Н.Ю. Прокопенко

ФУНКЦИОНАЛЬНЫЕ СИСТЕМЫ В ДИСКРЕТНОЙ МАТЕМАТИКЕ

Учебное пособие

Нижний Новгород
2024

Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Нижегородский государственный архитектурно-строительный университет»

Н.Ю. Прокопенко

ФУНКЦИОНАЛЬНЫЕ СИСТЕМЫ В ДИСКРЕТНОЙ МАТЕМАТИКЕ

Утверждено редакционно-издательским советом университета
в качестве учебного пособия

Нижний Новгород
ННГАСУ
2024

УДК 519(075)
П78
ББК 22.17я73

Публикуется в авторской редакции

Рецензенты:
О.В. Чермных – канд. физ.-мат. наук, доцент кафедры фундаментальной математики ФГБОУ ВО «Вятский государственный университет»

Е.М. Ковязина – канд. физ.-матем. наук, доцент, педагог КОГАОУ ДО «Центр дополнительного образования одаренных школьников»

Прокопенко, Н.Ю. Функциональные системы в дискретной математике : учебное пособие / Н.Ю. Прокопенко ; Министерство образования и науки Российской Федерации, Нижегородский государственный архитектурно-строительный университет. – Нижний Новгород : ННГАСУ, 2024. – 115 с. – ISBN 978-5-528-00573-7. – 1 CD ROM. – Текст : электронный.

Учебное пособие предназначено для обучающихся по очной форме в ННГАСУ по дисциплине «Дискретная математика» по направлению подготовки 09.03.03 Прикладная информатика и 09.03.04 Программная инженерия. Каждый раздел начинается с изложения необходимого теоретического материала, затем приводятся и разбираются примеры. Дается достаточное количество упражнений для самостоятельного решения.

ISBN 978-5-528-00573-7

Оглавление

Введение

Теория функциональных систем занимается изучением функций, описывающих работу дискретных преобразователей. К важнейшим классам функций относятся булевы функции, автоматные и вычислимые функции. С каждым из этих классов связываются операции, позволяющие из одних функций данного класса строить другие функции этого же класса. Такими операциями являются операция суперпозиции, операция обратной связи, операция примитивной рекурсии и μ-операция. В результате имеем функциональные системы с операциями – некоторые классы алгебр.

Роль функциональных систем в дискретной математике можно сравнить с ролью математического анализа в непрерывной математике.

Булевы функции названы в честь английского математика XIX века Дж. Буля, который впервые применил алгебраические методы для решения логических задач. Первоначально булевы функции рассматривались как логические формулы, были эффективным средством описания и решения различных логических задач. Булева алгебра стала математическим аппаратом для исследования релейно-контактных схем, а сами схемы к середине двадцатого века нашли многочисленные применения в автоматической технике, централизации и блокировке, релейной защите, телемеханике, при проектировании быстродействующих компьютеров. Помимо того, что булевы функции стали признанной моделью для проектирования схем, применяемых в электронике, во второй половине двадцатого века был открыт еще ряд важных применений теории булевых функций в таких областях, как распознавание образов, теория кодирования и криптография.

Многие результаты, полученные с помощью функциональных дискретных систем, легли в основу проектирования и создания компьютеров и программного обеспечения к ним, нашли широчайшее применение в различных областях информатики и систем искусственного интеллекта.

Раздел 1. Булевы функции

1.1. Дизъюнктивные и конъюнктивные нормальные формы

Определение 1.1. *Функцией алгебры логики n переменных (или булевой функцией)* называется любая функция n переменных $f(x_1, x_2, \ldots, x_n)$, аргументы которой принимают два значения 1 и 0, и при этом сама функция может принимать одно из двух значений 0 или 1.

Для задания булевой функции $f(\tilde{x}^n)$ требуется указать ее значения на каждом наборе E_2^n. При $n \geq 1$ функцию $f(\tilde{x}^n)$ можно задать таблицей T_f (табл. 1.1), называемой *таблицей истинности* функции, в которой наборы $\tilde{\sigma} = (\sigma_1, \ldots, \sigma_n)$ выписываются в порядке возрастания их номеров (сверху вниз).

Табл. 1.1.

x_1	x_2	…	x_{n-1}	x_n	$f(\tilde{x}^n)$
0	0	…	0	0	$f(0,0,\ldots,0,0)$
0	0	…	1	1	$f(0,0,\ldots,0,1)$
0	0	…	1	0	$f(0,0,\ldots,1,0)$
…	…	…	…	…	…
0	1	…	1	1	$f(0,1,\ldots,1,1)$
1	1	…	1	1	$f(1,1,\ldots,1,1)$

Всякая формула алгебры логики есть функция алгебры логики. Тождественно истинная и тождественно ложная формулы представляют собой постоянные функции, а две равносильные формулы выражают одну и ту же функцию.

Обозначим через $P_2(n)-$ множество всех булевых функций от n переменных, а $P_2 -$ множество всех булевых функций.

Используя правила комбинаторики, можно установить, что число различных функций алгебры логики n переменных равно числу двоичных векторов длины 2^n, т.е. 2^{2^n}.

Теорема. $|P_2(n)| = 2^{2^n}$.

Доказательство. Поскольку функция $f(\tilde{x}^n)$ задается вектором значений $\tilde{\alpha}_f = (\alpha_0, \alpha_1, \ldots, \alpha_{2^n-1})$, где $\alpha_i \in \{0, 1\}$, $0 \leq i \leq 2^n - 1$, то

различных булевых функций от n переменных столько же, сколько различных 2^n-разрядных двоичных векторов, т.е. 2^{2^n}.

Найдем все булевы функции одной переменной $y=f(x)$. Перенумеруем эти функции (их 4) естественным образом и представим в виде таблицы:

x	f_0	f_1	f_2	f_3
0	0	0	1	1
1	0	1	0	1

Видно, что $f_0(x) = 0$, а $f_3(x) = 1$, т. е. эти две функции не зависят от x,

$f_1(x)=x$, т. е. она не меняет аргумента.

Функция $f_2(x)$ принимает значения, противоположные значениям аргумента, обозначается $f_2(x)=\overline{x}$.

Найдем все булевы функции двух переменных $f(x_1,x_2)$.

Число этих функций равно $2^4 = 16$. Перенумеруем и расположим их в естественном порядке в таблице 1.2:

Табл. 1.2

x_1	x_2	f_0	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}	f_{11}	f_{12}	f_{13}	f_{14}	f_{15}
0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1	0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

Две функции $f_0(x,y) = 0$ и $f_{15}(x,y) = 1$ являются константами.

$f_1(x,y) = x \wedge y$, $f_2(x,y) = \overline{x \to y}$, $f_3(x,y) = x$; $f_4(x,y) = \overline{y \to x}$, $f_5(x,y) = y$, $f_6(x,y) = x \oplus y$, $f_7(x,y) = x \vee y$, $f_8(x,y) = x \downarrow y$ – стрелка Пирса, $f_9(x,y) = \overline{x \oplus y}$, $f_{10}(x,y) = \overline{y}$, $f_{11}(x,y) = y \to x$, $f_{12}(x,y) = \overline{x}$, $f_{13}(x,y) = x \to y$, $f_{14}(x,y) = x|y$ – штрих Шеффера.

Символы из множества $B = \{\neg, \wedge, \vee, \oplus, \leftrightarrow, \to, \downarrow, |\}$, в алгебре логики, участвующие в обозначениях элементарных функций, называют *логическими связками*.

Определение 1.2. Переменная x_i для функции $f(x_1, \dots, x_n)$ называется **существенной** (f зависит от x_i существенно), если существует такой набор $(\beta_1, \dots, \beta_{i-1}, \beta_{i+1}, \dots, \beta_n)$, что $f(\beta_1, \dots, \beta_{i-1}, 0, \beta_{i+1}, \dots, \beta_n) \neq f(\beta_1, \dots, \beta_{i-1}, 1, \beta_{i+1}, \dots, \beta_n)$. В противном случае переменная x_i – *фиктивная*, т.е. функция f не зависит от x_i.

Пример 1.1. Даны функции:

1) $f(a,b,c) = (b \to c \vee a) \wedge (a \to c \vee b)$

2) $f(a,b,c) = \bar{b} \wedge c \vee b \wedge (a \downarrow c \vee a \wedge \bar{c})$

3) $f(a,b,c) = (a \to b \vee c) \wedge (a \oplus c \vee \bar{b})$

4) $f(a,b,c) = \bar{a} \wedge (b \leftrightarrow c) \vee a \wedge b \wedge c)$

5) $f(a,b,c) = (\bar{a} | (b \downarrow c)) \wedge (a \to c \vee b)$

Проверим для каких функций переменная *a* является фиктивной.

Решение. Рассмотрим значения функций на наборах, которые отличаются только значением переменной *a* (на соседних наборах по переменной *a*).

1) f(0, 0, 0) = 1, f(0, 0, 1) = 1, f(0, 1, 0) = 0, f(0, 1, 1) = 1,
f(1, 0, 0) = 1, f(1, 0, 1) = 1, f(1, 1, 0) = 0, f(1, 1, 1) = 1.

Изменение значения переменной *a* в любом наборе значений переменных не изменяет значение функции, поэтому переменная *a* для этой функции является фиктивной.

2) f(0, 0, 0) = 0, f(0, 0, 1) = 1, f(0, 1, 0) = 1, f(0, 1, 1) = 0,
f(1, 0, 0) = 0, f(1, 0, 1) = 1, f(1, 1, 0) = 1, f(1, 1, 1) = 0.

Изменение значения переменной *a* в любом наборе значение переменных не изменяет значение функции, поэтому переменная *a* для этой функции является фиктивной.

3) f(0, 0, 0) = 1, f(1, 0, 0) = 0.

Изменение значения переменной *a* в наборе (0, 0, 0) приводит к изменению значения функции, поэтому переменная *a* для этой функции является существенной.

4) f(0, 0, 0) = 1, f(1, 0, 0) = 0.

Изменение значения переменной a в наборе (0, 0, 0) приводит к изменению значения функции, поэтому переменная a для этой функции является существенной.

5) f(0, 0, 0) = 0, f(0, 0, 1) = 1, f(0, 1, 0) = 1, f(0, 1, 1) = 1,

f(1, 0, 0) = 0, f(1, 0, 1) = 1, f(1, 1, 0) = 1, f(1, 1, 1) = 1.

Изменение значения переменной a в любом наборе значение переменных не изменяет значение функции, поэтому переменная a для этой функции является фиктивной. Итак, переменная a является фиктивной в переключательных функциях 1, 2, 5.■

Пример 1.2. Покажем, что x_1 – фиктивная переменная функции, реализовав для этой цели функцию формулой $f(x_1, x_2) = (x_1 \leftrightarrow x_2) \vee (x_1|x_2)$, не содержащей явно переменную x_1.

Решение. Упростим выражение для функции:

$$f(x_1, x_2) = (x_1 \leftrightarrow x_2) \vee (x_1|x_2) = (x_1 \wedge x_2) \vee (\bar{x}_1 \wedge \bar{x}_2) \vee (\bar{x}_1 \vee \bar{x}_2) =$$

$$= x_1 \wedge x_2 \vee \bar{x}_1 \wedge \bar{x}_2 \vee \bar{x}_1 \vee \bar{x}_2 = x_1 \wedge x_2 \vee \bar{x}_1 \vee \bar{x}_2 = (x_1 \vee \bar{x}_1) \wedge (x_2 \vee \bar{x}_1) \vee \bar{x}_2 =$$

$$= 1 \wedge (x_2 \vee \bar{x}_1) \vee \bar{x}_2 = x_2 \vee \bar{x}_1 \vee \bar{x}_2 = 1$$

Можно показать, что всякую булеву функцию можно представить в виде формулы алгебры логики следующим образом:

$$F(x_1, x_2, \ldots, x_n) = F(1,\ldots,1) \wedge x_1 \wedge x_2 \wedge \ldots \wedge x_n \vee F(1,\ldots,1,0) \wedge x_1 \wedge x_2 \wedge \ldots \wedge x_{n-1} \wedge \overline{x_n} \vee \vee F(1,\ldots,1,0,0) \wedge x_1 \wedge \ldots \wedge x_{n-2} \wedge \overline{x_{n-1}} \wedge \overline{x_n} \vee \ldots \vee F(0,0,\ldots,0) \wedge \overline{x_1} \wedge \overline{x_2} \wedge \ldots \wedge \overline{x_n} \quad (1)$$

или в виде формулы:

$$F(x_1, x_2, \ldots, x_n) = (F(1,\ldots,1) \vee \overline{x_1} \vee \ldots \vee \overline{x_n}) \wedge (F(1,\ldots,1,0) \vee \overline{x_1} \vee \ldots \vee \overline{x_{n-1}} \vee x_n) \wedge \quad \wedge (F(1,\ldots,1,0,0) \vee \overline{x_1} \vee \ldots \vee \overline{x_{n-2}} \vee x_{n-1} \vee x_n) \wedge \ldots \wedge (F(0,\ldots,0) \vee x_1 \vee \ldots \vee x_n) \quad (2)$$

Формулы (1) и (2) можно привести при помощи равносильных преобразований в алгебре высказываний к некоторому специальному виду – *совершенной нормальной форме*.

Определение 1.3. *Элементарной конъюнкцией n* переменных называется конъюнкция переменных и их отрицаний.

Примеры элементарных конъюнкций: $x \wedge \bar{y} \wedge \bar{x} \wedge z$, $x \wedge y \wedge \bar{z}$, $y \wedge \bar{y} \wedge \bar{x}$.

Элементарной дизъюнкцией n переменных называется дизъюнкция переменных и их отрицаний.

Примеры элементарных дизъюнкций: $y \vee \bar{y} \vee \bar{z} \vee x$, $x \vee y \vee \bar{x}$, $x \vee z \vee x$.

Определение 1.4. *Дизъюнктивной нормальной формой* (ДНФ) формулы *A* называется равносильная ей формула, представляющая собой дизъюнкцию элементарных конъюнкций.

Конъюнктивной нормальной формой (КНФ) формулы *A* называется равносильная ей формула, представляющая собой конъюнкцию элементарных дизъюнкций.

Для любой формулы алгебры логики путем равносильных преобразований можно получить ДНФ и КНФ, причем не единственную.

Определение 1.5. *Совершенной дизъюнктивной нормальной формой* (СДНФ) формулы *A* называется ДНФ *A*, обладающая следующими свойствами:

1. Все элементарные конъюнкции, входящие в ДНФ *A*, различны.

2. Все элементарные конъюнкции, входящие в ДНФ *A*, содержат все переменные, участвующие в формуле.

3. Каждая элементарная конъюнкция, входящая в ДНФ *A*, не содержит двух одинаковых выражений.

4. Каждая элементарная конъюнкция не содержит одновременно переменную и ее отрицание.

СДНФ для формулы единственна с точностью до перестановки дизъюнктивных и конъюнктивных членов.

СДНФ *A* можно получить двумя способами:

а) с помощью таблицы истинности;

б) с помощью равносильных преобразований.

Построение СДНФ с помощью таблицы истинности:

1. Составить таблицу истинности данной логической формулы или булевой функции.

2. Указать в таблице строки, где формула равна 1.

3. Для каждого набора значений переменных, на котором формула имеет значение 1, выписать конъюнкции всех переменных, причем над теми переменными, которые на этом наборе равны 0, ставятся отрицания.

4. Все полученные конъюнкции нужно соединить знаками дизъюнкции.

Пример 1.3. Булеву функцию трех переменных

$F(x_1, x_2, x_3) = (x_1 \leftrightarrow \overline{x_2}) \rightarrow (x_1 \vee x_3) \wedge x_2$ представить логической формулой – в виде СДНФ.

x_1 x_2 x_3	$\overline{x_2}$	$x_1 \leftrightarrow \overline{x_2}$	$x_1 \vee x_3$	$(x_1 \vee x_3) \wedge x_2$	$F(x_1, x_2, x_3)$	
0 0 0	1	0	0	0	1	$\overline{x_1} \wedge \overline{x_2} \wedge \overline{x_3}$
0 0 1	1	0	1	0	1	$\overline{x_1} \wedge \overline{x_2} \wedge x_3$
0 1 0	0	1	0	0	0	
0 1 1	0	1	1	1	1	$\overline{x_1} \wedge x_2 \wedge x_3$
1 0 0	1	1	1	0	0	
1 0 1	1	1	1	0	0	
1 1 0	0	0	1	1	1	$x_1 \wedge x_2 \wedge \overline{x_3}$
1 1 1	0	0	1	1	1	$x_1 \wedge x_2 \wedge x_3$

Искомая СДНФ логической функции

$$F(x_1, x_2, x_3) = \overline{x_1} \wedge \overline{x_2} \wedge \overline{x_3} \vee \overline{x_1} \wedge \overline{x_2} \wedge x_3 \vee \overline{x_1} \wedge x_2 \wedge x_3 \vee x_1 \wedge x_2 \wedge \overline{x_3} \vee x_1 \wedge x_2 \wedge x_3$$

Построение СДНФ *A* с помощью равносильных преобразований:

1. Для формулы *A* получить любую ДНФ (пусть $A \equiv B_1 \vee B_2 \vee \ldots \vee B_к$).

2. Если *B* есть элементарная конъюнкция, не содержащая переменную x_i, то нужно заменить *B* равносильной формулой:

$B \wedge (x_i \vee \overline{x_i}) \equiv B \wedge x_i \vee B \vee \overline{x_i}$..

3. Если в ДНФ есть два одинаковых выражения $B \vee B$, то одно можно отбросить, так как $B \vee B \equiv B$.

4. Если в некоторую элементарную конъюнкцию B переменная x_i входит дважды, то лишнюю переменную надо отбросить, так как $x_i \wedge x_i = x_i$.

5. Если B содержит конъюнкцию $x_i \wedge \overline{x_i}$, то это слагаемое можно отбросить, так как $x_i \wedge \overline{x_i} \equiv 0$, и, следовательно, $B \equiv 0$, а ложное высказывание из дизъюнкции можно отбросить в силу равносильности $C \vee 0 \equiv C$.

Пример 1.4. Для формулы $A \equiv x \vee y \wedge (x \vee \overline{y})$ построить СДНФ.

Решение. ДНФ $A \equiv x \vee y \wedge (x \vee \overline{y})$

$A \equiv x \vee y \wedge (x \vee \overline{y}) \equiv x \vee y \wedge x \vee 0 \equiv x \equiv x \wedge (y \vee \overline{y}) \equiv x \wedge y \vee x \wedge \overline{y}$

СДНФ $A \equiv x \wedge y \vee x \wedge \overline{y}$.

Определение 1.6. КНФ A называется *совершенной конъюнктивной нормальной формой* формулы A (СКНФ), если для нее выполнены условия:

1. Все элементарные дизъюнкции, входящие в КНФ A, различны.

2. Все элементарные дизъюнкции, входящие в КНФ A, содержат все переменные, участвующие в формуле.

3. Каждая элементарная дизъюнкция, входящая в КНФ A, не содержит двух одинаковых выражений.

4. Каждая элементарная дизъюнкция не содержит одновременно переменную и ее отрицание.

Можно доказать, что каждая не тождественно истинная формула имеет единственную СКНФ. СКНФ A можно получить двумя способами: а) с помощью таблицы истинности (используя закон двойственности $\text{СКНФ}A \equiv \overline{\text{СДНФ}\overline{A}}$); б) с помощью равносильных преобразований.

Построение СКНФ с помощью таблицы истинности:

1. Составить таблицу истинности данной логической формулы или булевой функции.

2. Указать в таблице строки, где формула равна 0.

3. Для каждого набора значений переменных, на котором формула имеет значение 0, выписать дизъюнкции всех переменных, причем отрицание ставится над теми переменными, которые на этом наборе имеют значение 1.

4. Все полученные дизъюнкции нужно соединить знаками конъюнкции.

Пример 1.5. Булеву функцию трех переменных

$F(x_1, x_2, x_3) = (x_1 \leftrightarrow \overline{x_2}) \rightarrow (x_1 \vee x_3) \wedge x_2$ представить логической формулой – в виде СКНФ.

x_1 x_2 x_3	$\overline{x_2}$	$x_1 \leftrightarrow \overline{x_2}$	$x_1 \vee x_3$	$(x_1 \vee x_3) \wedge x_2$	$F(x_1, x_2, x_3)$	
0 0 0	1	0	0	0	1	
0 0 1	1	0	1	0	1	
0 1 0	0	1	0	0	0	$x_1 \vee \overline{x_2} \vee x_3$
0 1 1	0	1	1	1	1	
1 0 0	1	1	1	0	0	$\overline{x_1} \vee x_2 \vee x_3$
1 0 1	1	1	1	0	0	$\overline{x_1} \vee x_2 \vee \overline{x_3}$
1 1 0	0	0	1	1	1	
1 1 1	0	0	1	1	1	

Искомая СКНФ логической функции:

$$F(x_1, x_2, x_3) = (x_1 \vee \overline{x_2} \vee x_3) \wedge (\overline{x_1} \vee x_2 \vee x_3) \wedge (\overline{x_1} \vee x_2 \vee \overline{x_3})$$

Построение СКНФ с помощью равносильных преобразований:

1. Для формулы A получить любую КНФ (пусть $A \equiv B_1 \wedge B_2 \wedge \ldots \wedge B_к$).

2. Если B есть элементарная дизъюнкция, не содержащая переменную x_i, то нужно заменить B равносильной формулой:

$B \vee x_i \wedge \overline{x_i} \equiv (B \vee x_i) \wedge (B \vee \overline{x_i})$.

3. Если в КНФ есть два одинаковых выражения $B \wedge B$, то одно можно отбросить, так как $B \wedge B \equiv B$.

4. Если в некоторую элементарную дизъюнкцию B переменная x_i входит дважды, то лишнюю переменную надо отбросить, так как $x_i \vee x_i \equiv x_i$.

5. Если B содержит дизъюнкцию $x_i \vee \overline{x_i}$, то это слагаемое можно отбросить, так как $x_i \vee \overline{x_i} \equiv 1$, и, следовательно, $B \equiv 1$, а истинное высказывание из конъюнкции можно отбросить в силу равносильности $C \wedge 1 \equiv C$.

Пример 1.6. Для формулы $A \equiv x \vee y \wedge (x \vee \overline{y})$ построить СКНФ.

Решение. КНФ $A \equiv (x \vee y) \wedge (x \vee x \vee \overline{y})$, СКНФ $A \equiv (x \vee y) \wedge (x \vee \overline{y})$.

Пример 1.7. Для следующих формул найдите СДНФ и СКНФ, каждую двумя способами (путем равносильных преобразований и, используя таблицы истинности): 1) $x \wedge (x \to y)$; 2) $(\overline{x \wedge y} \to \bar{x}) \wedge \overline{x \wedge y \to \bar{y}}$; 3) $(x \to y) \to (y \to x)$.

Решение.

1) $x \wedge (x \to y) \equiv x \wedge (\bar{x} \vee y) \equiv x \wedge \bar{x} \vee x \wedge y) \equiv x \wedge y$ – является ДНФ, СДНФ, КНФ, но не является СКНФ.

   $x \wedge y \equiv (x \vee y \wedge \bar{y}) \wedge (y \vee x \wedge \bar{x}) \equiv (x \vee y) \wedge (x \vee \bar{y}) \wedge (y \vee x) \wedge (y \vee \bar{x}) \equiv$ $(x \vee y) \wedge (x \vee \bar{y} \wedge (y \vee \bar{x})$ – СКНФ.

2) $(\overline{x \wedge y} \to \bar{x}) \wedge \overline{x \wedge y \to \bar{y}} \equiv (x \wedge y \vee \bar{x}) \wedge (x \wedge y \wedge y) \equiv (x \vee \bar{x}) \wedge (y \vee \bar{x}) \wedge$ $(x \wedge y) \equiv (y \vee \bar{x}) \wedge x \wedge y) \equiv y \wedge x \wedge y \vee \bar{x} \wedge x \wedge y \equiv y \wedge x$ (далее, как в примере 1)

3) $(x \to y) \to (y \to x) \equiv \overline{\bar{x} \vee y} \vee \bar{y} \vee x \equiv x \wedge \bar{y} \vee \bar{y} \vee x \equiv \bar{y} \vee x$ – является КНФ, СКНФ, ДНФ, но не является СДНФ.

   $\bar{y} \wedge (x \vee \bar{x}) \vee x \wedge (y \vee \bar{y}) \equiv \bar{y} \wedge x \vee \bar{y} \wedge \bar{x} \vee x \wedge y \vee x \wedge \bar{y} \equiv \bar{y} \wedge x \vee \bar{y} \wedge \bar{x} \vee$ $x \wedge y$ – является СДНФ.

1.2. Сокращенные ДНФ

Сокращенная ДНФ являются еще одним способом однозначного представления булевых функций, которое во многих случаях может оказаться более простым, чем представление с помощью СДНФ.

Определение 1.7. *Сокращенная* ДНФ – форма записи функции, обладающая следующими свойствами: любые два слагаемых различаются как минимум в двух позициях, ни один из конъюнктов не содержится в другом.

Метод Нельсона позволяет получать сокращенную ДНФ булевой функции f из ее произвольной конъюнктивной нормальной формы. Суть метода заключается в использовании следующего утверждения, которое мы приводим без доказательства: если в произвольной КНФ булевой функции f

раскрыть скобки и произвести все поглощения, то в результате будет получена сокращенная ДНФ булевой функции *f*.

Пример 1.8. Булева функция f задана как ДНФ

$$f = (x_1 \vee \bar{x}_2)(\bar{x}_1 \vee x_3)(x_1 \vee x_2 \vee \bar{x}_3).$$

Найти методом Нельсона сокращенную ДНФ функции *f*.

После раскрытия скобок получаем:

$$f = (x_1x_3 \vee \bar{x}_1\bar{x}_2 \vee \bar{x}_2x_3)(x_1 \vee x_2 \vee \bar{x}_3) = x_1x_3 \vee x_1x_2x_3 \vee \bar{x}_1\bar{x}_2\bar{x}_3 \vee x_1\bar{x}_2x_3.$$

После проведения всех поглощений имеем $f = x_1x_3 \vee \bar{x}_1\bar{x}_2\bar{x}_3$. Получена сокращенная ДНФ функции *f*.

Метод Блейка-Порецкого позволяет получать сокращенную ДНФ булевой функции f из ее произвольной ДНФ. Базируется на применении формулы обобщенного склеивания: $Ax \vee B\bar{x} = Ax \vee B\bar{x} \vee AB$, справедливость которой легко доказать. Действительно, $Ax = ABx$; $B\bar{x} = B\bar{x} \vee AB\bar{x}$.

Следовательно, $Ax \vee B\bar{x} = Ax \vee ABx \vee B\bar{x} \vee AB\bar{x} = Ax \vee B\bar{x} \vee AB$.

В основу метода положено следующее утверждение: если в произвольной ДНФ булевой функции f произвести все возможные обобщенные склеивания, а затем выполнить все поглощения, то в результате получится сокращенная ДНФ функции *f*.

Метод Квайна основывается на применении двух основных соотношений:

1. Соотношение склеивания $x \wedge K \vee \wedge \bar{x}K = K$.

Закон неполного склеивания: $Ax \vee A\bar{x} = Ax \vee A\bar{x} \vee A$,

Закон обобщенного склеивания: $x\,K_1 \vee x\,K_2 = x\,K_1 \vee \bar{x}K_2 \vee K_1K_2$.

где A – любое элементарное произведение.

2. Соотношение поглощения $A\tilde{x} \vee A = A, \tilde{x} \in \{x, \bar{x}\}$.

Пример 1.9. Булева функция *f* задана произвольной ДНФ.

$$f = \bar{x}_1\bar{x}_2 \vee x_1\bar{x}_2\bar{x}_3 \vee x_1x_2.$$

Найти методом Блейка-Порецкого сокращенную ДНФ функции *f*. Проводим обобщенные склеивания. Легко видеть, что первый и второй элемент заданной ДНФ допускают обобщенное склеивание по переменной x_1. В результате склеивания имеем:

$$\bar{x}_1\bar{x}_2 \vee x_1\bar{x}_2\bar{x}_3 = \bar{x}_1\bar{x}_2 \vee x_1\bar{x}_2\bar{x}_3 \vee \bar{x}_2\bar{x}_3 .$$

Первый и третий элемент исходной ДНФ допускают обобщенное склеивание как. по переменной x_1так и поx_2. После склеивания по x_1 имеем:

$$\bar{x}_1\bar{x}_2 \vee x_1x_2 = \bar{x}_1\bar{x}_2 \vee x_1x_2 \vee \bar{x}_2x_2 = \bar{x}_1\bar{x}_2 \vee x_1x_2 .$$

После склеивания по x_2 имеем:

$$\bar{x}_1\bar{x}_2 \vee x_1x_2 = \bar{x}_1\bar{x}_2 \vee x_1x_2 \vee \bar{x}_1x_1 = \bar{x}_1\bar{x}_2 \vee x_1x_2 .$$

Второй и третий элемент ДНФ допускают обобщенное склеивание по переменной x_2 . После склеивания получаем:

$$x_1\bar{x}_2\bar{x}_3 \vee x_1x_2 = x_1\bar{x}_2\bar{x}_3 \vee x_1x_2 \vee x_1x_3 .$$

Выполнив последнее обобщенное склеивание, приходим к ДНФ:

$$f = \bar{x}_1\bar{x}_2 \vee x_1\bar{x}_2\bar{x}_3 \vee \bar{x}_2\bar{x}_3 \vee x_1x_2 \vee x_1\bar{x}_3 .$$

После выполнения поглощений получаем $f = \bar{x}_1\bar{x}_2 \vee \bar{x}_2\bar{x}_3 \vee x_1x_2 \vee x_1\bar{x}_3$.

Попытки дальнейшего применения операции обобщенного склеивания и поглощения не дают результата. Следовательно, получена сокращенная ДНФ функции *f*.

1.3. Многочлены Жегалкина

Определение 1.8. *Многочленом (полиномом) Жегалкина* от *n* переменных называется функция

$$f(x_1, x_2, \dots, x_n) \equiv a_0x_1x_2 \dots x_n \oplus a_1x_1x_2 \dots x_{n-1} \oplus \dots \oplus a_{m-1}x_n \oplus a_m$$

Всего здесь 2^n слагаемых. Напомним, что $\oplus$ означает сложение по модулю 2, коэффициенты a_i являются константами (равными нулю или единице).

Теорема. Любая функция *n* переменных может быть представлена полиномом Жегалкина и это представление единственно.

Функция $f(x_1, x_2,\ldots, x_n)$ называется *линейной*, если ее полином Жегалкина содержит только первые степени слагаемых. Более точно функция называется линейной, если ее можно представить в виде:

$$f(x_1, x_2, \ldots, x_n) \equiv a_0 \oplus a_1 x_1 \oplus \ldots \oplus a_n x_n$$

Построение полинома Жегалкина с помощью таблицы истинности:

1. Составить таблицу истинности данной логической формулы или булевой функции.

2. Указать в таблице строки, где формула равна 1.

3. Для каждого набора значений переменных, на котором формула имеет значение 1, выписать конъюнкции всех переменных, причем над теми переменными, которые на этом наборе равны 0, ставятся отрицания.

4. Все полученные конъюнкции нужно соединить знаками $\oplus$ суммы по модулю 2.

5. Все отрицания заменяем равносильной формулой $\bar{x} \equiv x \oplus 1$, раскрываем скобки и упрощаем, используя формулу: $x \oplus x \equiv 0$.

Пример 1.10. Представим в виде полинома Жегалкина дизъюнкцию $f(x_1, x_2) \equiv x_1 \vee x_2$.

x_1	x_2	$x_1 \vee x_2$	
0	0	0	
0	1	1	$\overline{x_1} \wedge x_2$
1	0	1	$x_1 \wedge \overline{x_2}$
1	1	1	$x_1 \wedge x_2$

$f(x_1, x_2) \equiv x_1 \vee x_2 \equiv \overline{x_1} \wedge x_2 \oplus x_1 \wedge \overline{x_2} \oplus x_1 \wedge x_2 \equiv (x_1 \oplus 1) \wedge x_2 \oplus x_1 \wedge (x_2 \oplus 1) \oplus x_1 \wedge x_2 \equiv x_1 \wedge x_2 \oplus x_2 \oplus x_1 \wedge x_2 \oplus x_1 \oplus x_1 \wedge x_2 \equiv x_1 \wedge x_2 \oplus x_1 \oplus x_2$.

Алгоритм построения полинома Жегалкина с помощью метода неопределенных коэффициентов:

Метод неопределенных коэффициентов состоит в следующем: записываем булеву функцию в виде полинома Жегалкина с неопределенными коэффициентами. Приравниваем значения функции к значениям полинома на соответствующих наборах переменных и находим неизвестные коэффициенты.

Пример 1.11. Для функции, заданной своими значениями:

$f = (11001011)$, построим полином методом неопределённых коэффициентов.

Для этого запишем нашу функцию в виде многочлена с неопределенными коэффициентами

x	y	z	f	$= a_0 \oplus a_1 x \oplus a_2 y \oplus a_3 z \oplus a_{12} xy \oplus a_{13} xz \oplus a_{23} yz \oplus a_{123} xyz$	a
0	0	0	1	$= a_0$	$a_0 = 1$
0	0	1	1	$= a_0 \oplus a_3 = 1 \oplus a_3$	$a_3 = 0$
0	1	0	0	$= a_0 \oplus a_2 = 1 \oplus a_2$	$a_2 = 1$
0	1	1	0	$= a_0 \oplus a_2 \oplus a_3 \oplus a_{23} = 1 \oplus 1 \oplus 0 \oplus a_{23}$	$a_{23} = 0$
1	0	0	1	$= a_0 \oplus a_1 = 1 \oplus a_1$	$a_1 = 0$
1	0	1	0	$= a_0 \oplus a_1 \oplus a_3 \oplus a_{13} = 1 \oplus 0 \oplus 0 \oplus a_{13}$	$a_{13} = 1$
1	1	0	1	$= a_0 \oplus a_1 \oplus a_2 \oplus a_{12} = 1 \oplus 0 \oplus 1 \oplus a_{12}$	$a_{12} = 1$
1	1	1	1	$= a_0 \oplus a_1 \oplus a_2 \oplus a_3 \oplus a_{12} \oplus a_{13} \oplus a_{23} \oplus a_{123} = 1 \oplus 0 \oplus 1 \oplus 0 \oplus 1 \oplus 1 \oplus 0 \oplus a_{123}$	$a_{123} = 1$

Подставляя найденные значения a_i, в формулу, получим полином Жегалкина $P = 1 \oplus y \oplus xy \oplus xz \oplus xyz$.

Алгоритм построения полинома Жегалкина с помощью равносильных преобразований:

1. Минимизируем булеву функцию любым известным нам способом.
2. Заменяем дизъюнкцию суммой по модулю 2: $f_1 \vee f_2 = f_1 \oplus f_2 \oplus f_1 f_2$.
3. Заменяем $\bar{x_i} = x_i \oplus 1$.
4. Используем распределительный закон (раскрываем скобки)
5. Применяем $f \oplus f = 0$ и $f \oplus 0 = f$.

Пример 1.12. Функцию $f = (x \vee y) \wedge (\bar{y} \vee xz)$ преобразуем с помощью равносильных преобразований.

$$f = (x \vee y) \wedge (\bar{y} \vee xz) = (x \oplus y \oplus xy)((y \oplus 1) \oplus xz \oplus (y \oplus 1)xz)$$
$$= (x \oplus y \oplus xy)((y \oplus 1 \oplus xz \oplus xz \oplus xyz) = (x \oplus y \oplus xy)((y \oplus 1 \oplus xyz)$$
$$= xy \oplus x \oplus xyz \oplus y \oplus y \oplus xyz \oplus xy \oplus xy \oplus xyz = x \oplus xy \oplus xyz$$

1.4. Проблема разрешимости

Все формулы алгебры высказываний делятся на три класса: тождественно истинные, тождественно ложные, выполнимые.

Проблема разрешимости может быть сформулирована следующим образом: существует ли способ, который позволял бы для каждой формулы алгебры высказываний за конечное число шагов ответить на вопрос, к какому классу эта формула принадлежит.

Очевидно, проблема разрешимости алгебры высказываний разрешима. Действительно, для каждой логической формулы может быть записана таблица истинности, которая и даст ответ на поставленный вопрос. Однако практическое использование таблицы истинности для формулы $A(x_1, x_2, \dots, x_n)$ при больших *n* затруднительно.

Существует другой способ проверки к какому классу принадлежит данная формула, этот способ основан на приведении формулы к ДНФ и КНФ.

Критерий разрешимости. Для того, чтобы формула алгебры высказываний *A* была тождественна истинна (тождественно ложна), необходимо и достаточно, чтобы любая элементарная дизъюнкция (конъюнкция), входящая в КНФ (ДНФ) формулы *A*, содержала переменную и ее отрицание.

Пример 1.13. Проверим будет ли формула $\mathrm{A} \equiv (\mathrm{x} \to \mathrm{y}) \to \overline{\mathrm{x}} \wedge \mathrm{y} \vee \overline{\mathrm{y}}$ тождественно истинной, тождественно ложной или выполнимой.

Решение. Приведем формулу *A* к ДНФ.

$(x \to y) \to \overline{x} \wedge y \vee \overline{y} \equiv \overline{x} \vee y \to \overline{x} \wedge y \vee \overline{y} \equiv \overline{\overline{x} \vee y} \vee \overline{x} \wedge y \vee \overline{y} \equiv x \wedge \overline{y} \vee \overline{x} \wedge y \vee \overline{y}$.

Полученная ДНФ не является тождественно ложной, так как каждая элементарная конъюнкция не содержит переменную и ее отрицание. Следовательно, исходная формула тождественно истинна или выполнима.

Преобразуем данную формулу к КНФ: $(x \to y) \to \overline{x} \wedge y \vee \overline{y} \equiv$

$\equiv x \wedge \overline{y} \vee \overline{x} \wedge y \vee \overline{y} \equiv (x \wedge \overline{y} \vee \overline{y}) \vee \overline{x} \wedge y \equiv \overline{y} \vee \overline{x} \wedge y \equiv (\overline{y} \vee \overline{x}) \wedge (\overline{y} \vee y) \equiv \overline{y} \vee \overline{x}$.

Полученная КНФ не является тождественно истинной, так как элементарная дизъюнкция не содержит переменную и ее отрицание.

Следовательно, данная формула *A* выполнима.

Задачи

1. Запишите формулами все функции алгебры логики одной и двух переменных (составить их таблицы истинности и записать аналитические выражения этих функций).
2. Для следующих формул найдите СДНФ и СКНФ, каждую двумя способами (путем равносильных преобразований и, используя таблицы истинности):

   1) $x \wedge (x \to y)$;

   2) $(\overline{x \wedge y} \to \overline{x}) \wedge \overline{x \wedge y \to \overline{y}}$;

   3) $(x \to y) \to (y \to x)$;

   4) $(x \vee \overline{z}) \to y \wedge z$;

   5) $(\overline{a} \to c) \to \overline{\overline{b} \to \overline{a}}$;

   6) $(x \vee \overline{y} \to x \wedge z) \to \overline{x \to \overline{x}} \vee y \wedge \overline{z}$;

   7) $(\overline{a} \to \overline{b}) \to (b \wedge c \to a \wedge c)$;

   8) $(a \wedge b \to b \wedge c) \to ((a \to b) \to (c \to b))$;

   9) $x_1 \to (x_2 \to (\ldots \to (x_{n-1} \to x_n)\ldots))$;

   10) $x_1 \vee x_2 \vee \ldots \vee x_n \to y_1 \wedge y_2 \wedge \ldots \wedge y_n$.

3. Докажите равносильность формул $\overline{x \wedge \overline{y}} \to (\overline{y} \to x)$ и $\overline{x \to y} \vee x \vee y$ сравнением их совершенных нормальных форм (конъюнктивных или дизъюнктивных).
4. Укажите все фиктивные переменные у функции f:

   а) $f(\tilde{x}^3) = (10101010)$;

   б) $f(\tilde{x}^4) = (1011010110110101)$;

   в) $f(\tilde{x}^4) = (0101111101011111)$.
5. По таблицам истинности найдите формулы, определяющие функции $F_1(x,y,z)$, $F_2(x,y,z)$, $F_3(x,y,z)$, $F_4(x,y,z)$, и придайте им более простой вид:

x	y	z	$F_1(x,y,z)$	$F_2(x,y,z)$	$F_3(x,y,z)$	$F_4(x,y,z)$
0	0	0	0	0	0	0
0	0	1	1	0	1	1
0	1	0	0	1	1	0
0	1	1	0	0	0	0
1	0	0	1	0	0	1
1	0	1	1	0	0	1
1	1	0	1	1	1	0
1	1	1	0	1	1	1

6. Найдите более простой вид формул, имеющих следующие совершенные нормальные формы:

   1) $x \wedge y \vee x\overline{y} \vee \overline{x} \wedge y$;

   2) $x \wedge y \wedge z \vee \overline{x} \wedge y \wedge z \vee x \wedge \overline{y} \wedge z$;

   3) $(x \vee \overline{y}) \wedge (\overline{x} \vee y) \wedge (\overline{x} \vee \overline{y})$;

   4) $(x \vee y \vee \overline{z}) \wedge (\overline{x} \vee y \vee z) \wedge (x \vee \overline{y} \vee \overline{z})$.
7. Используя критерий тождественной истинности и тождественной ложности формулы, установите, будет ли данная формула тождественно истиной, тождественно ложной или выполнимой:

   1) $\overline{x \wedge \overline{y}} \leftrightarrow \overline{x} \vee x \wedge y$;

   2) $(x \leftrightarrow y) \wedge (x \wedge \overline{y} \vee \overline{x} \wedge y)$;

   3) $x \wedge y \to (x \to \overline{y})$;

   4) $x \vee y \to (x \leftrightarrow y)$;

5) $x \vee y \to z$;

6) $(x \to z) \wedge (y \to z) \wedge (x \to y)$.

8. Найти сокращенную ДНФ и многочлен Жегалкина (методом неопределенных коэффициентов) для следующих функций:

a) $f(\tilde{x}^3) = (0010\ 1100)$; b) $f(\tilde{x}^3) = (1100\ 0010)$;

c) $f(\tilde{x}^3) = (1110\ 1100)$; d) a) $f(\tilde{x}^3) = (0110\ 1011)$

9. Для следующих функций: записать полином Жегалкина (определить существенные и фиктивные переменные.

1) $x \vee y$

2) $x \to y$

3) $x \leftrightarrow y$

4) $x \downarrow y$

5) $x|y$

6) $x_1 \longrightarrow ((x_2 \vee x_3) \downarrow x_1)$

7) $f = (01010001)$

8) $(x_1 \downarrow (x_2 \leftrightarrow x_3)) \to x_2)$

9) $x_1 \leftrightarrow ((x_2 \leftrightarrow x_3) \to x_3)$

10. Показать, что x_1 – фиктивная переменная у функции

$a)\ f(x_1, x_2) = (x_2 \to x_1) \wedge (x_2 \downarrow x_2)$,

$b)\ f(x_1, x_2) = (x_2 \leftrightarrow x_1) \vee (x_1|x_2)$,

c) $f(x_1, x_2, x_3) = ((x_2 \oplus x_1) \to x_3) \wedge \overline{x_3 \to x_2}$,

реализовав для этой цели функцию формулой, не содержащей явно переменную, например найти полином Жегалкина).

1.5. Релейно-контактные схемы (РКС)

Релейно-контактные схемы (их часто называют переключательными схемами) широко используются в технике автоматного управления.

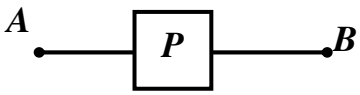
Определение 1.9. Под переключательной схемой понимают схематическое изображение некоторого устройства, состоящее из следующих элементов:

1) *переключателей*, которыми могут быть механические устройства, электромагнитные реле, полупроводники и т.д.;

2) соединяющие их *проводники*;

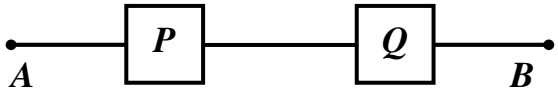
3) *входы* в схему и *выходы* из нее (клеммы, на которые подается электрическое напряжение). Они называются полюсами.

Простейшая схема содержит один переключатель P и имеет один вход A и один выход B. Переключателю P поставим в соответствие высказывание p: «Переключатель P замкнут». Если p истинно, то импульс, поступающий на полюс A, может быть снят на полюсе B без потери напряжения, т. е. схема пропускает ток. Если p ложно, то переключатель разомкнут, и схема тока не проводит. Таким образом, если принять во внимание не смысл высказывания, а только его значение, то можно считать, что любому высказыванию может быть поставлена в соответствие переключательная схема с двумя полюсами (двухполюсная схема).

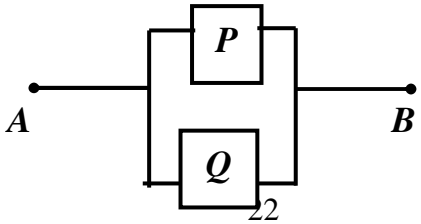


Формулам, включающим основные логические операции, также могут быть поставлены в соответствие переключательные схемы.

Так, конъюнкции двух высказываний $p \wedge q$ ставится в соответствие схема:



а дизъюнкции $p \vee q$ схема:

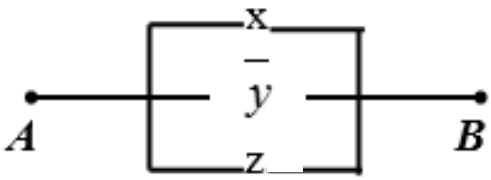


Так как любая формула алгебры логики может быть записана в виде ДНФ или КНФ, то ясно, что каждой формуле алгебры логики можно поставить в соответствие некоторую РКС, а каждой РКС можно поставить в соответствие некоторую формулу алгебры логики. Поэтому возможности схемы можно выявить, изучая соответствующую ей формулу, а упрощение схемы можно свести к упрощению формулы.

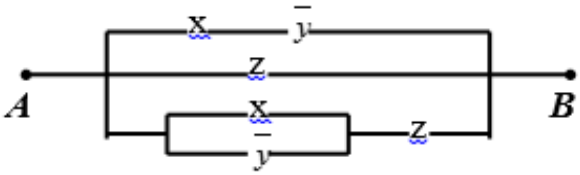
Пример 1.14. Составить РКС для формулы $(\bar{x} \wedge y) \rightarrow (z \vee x)$.

Решение. Упростим данную формулу с помощью равносильных преобразований: $(\bar{x} \wedge y) \rightarrow (z \vee x) \equiv \overline{\bar{x} \wedge y} \vee z \vee x \equiv x \vee \bar{y} \vee z \vee x \equiv x \vee \bar{y} \vee z$.

Тогда РКС для данной формулы имеет вид:



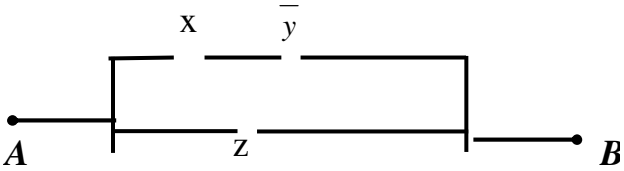
Пример 1.15. Упростить РКС



Решение. Составим по данной РКС формулу (функцию проводимости) и упростим ее: $(x \wedge \bar{y}) \vee z \vee (x \vee \bar{y}) \wedge z \equiv x \wedge \bar{y} \vee z$.

(к последним двум слагаемым применили закон поглощения).

Тогда упрощенная схема выглядит так:



Пример 1.16. (пример построения РКС по заданным условиям с оценкой числа контактов)

Построить контактную схему для оценки результатов некоторого спортивного соревнования тремя судьями при следующих условиях: судья, засчитывающий результат, нажимает имеющуюся в его распоряжении

кнопку, а судья, не засчитывающий результат, кнопки не нажимает. В случае, если кнопки нажали не менее двух судей, должна загореться лампочка положительное решение судей принято простым большинством голосов).

Решение. Ясно, что работа нужной РКС описывается булевой функцией трех переменных $F(x,y,z)$, где переменные высказывания x, y, z означают: x – судья x голосует «за»; y – судья y голосует «за»; z – судья z голосует «за».

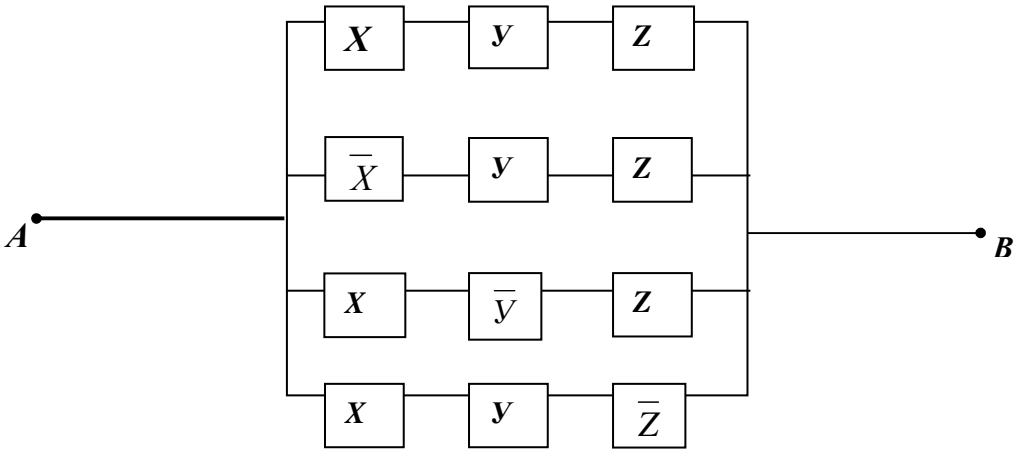
Таблица истинности функции $F(x,y,z)$, очевидно, имеет вид:

x	y	z	$F(x,y,z)$
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

СКНФ формулы (функции) $F(x,y,z)$ запишется в виде:

$F(x,y,z) \equiv x \wedge y \wedge z \vee x \wedge y \wedge \overline{z} \vee x \wedge \overline{y} \wedge z \vee \overline{x} \wedge y \wedge z.$

Этой формуле соответствует РКС с двенадцатью переключателями:



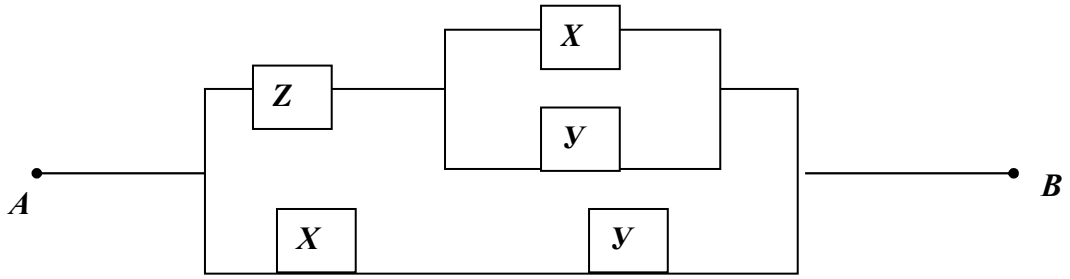
Упростим формулу $F(x,y,z) \equiv x \wedge y \wedge z \vee x \wedge y \wedge \overline{z} \vee x \wedge \overline{y} \wedge z \vee \overline{x} \wedge y \wedge z \equiv$

$\equiv (x \wedge y \wedge z \vee x \wedge y \wedge \overline{z}) \vee (\text{x} \wedge \text{y} \wedge z \vee x \wedge \overline{y} \wedge z) \vee (x \wedge y \wedge z \vee \overline{x} \wedge y \wedge z) \equiv$

$\equiv \text{x} \wedge \text{y} \wedge (z \vee \overline{z}) \vee x \wedge z \wedge (y \vee \overline{y}) \vee y \wedge z \wedge (x \vee \overline{x}) \equiv x \wedge y \vee x \wedge z \vee y \wedge z \equiv$

$\equiv x \wedge y \vee z \wedge (x \vee y).$

Полученной формуле соответствует схема, содержащая пять переключателей.



Задачи

1. Составьте РКС для формул:

1) $x \to y$; 2) $x \leftrightarrow y$;

3) $x \wedge (\overline{y} \wedge z \vee x \vee y)$; 4) $x \wedge y \wedge z \vee \overline{x \wedge y \wedge z} \vee \overline{x} \wedge y$;

5) $x \wedge (y \wedge z \vee \overline{y \wedge z}) \vee \overline{x} \wedge (\overline{y} \wedge z \vee y \wedge \overline{z})$; 6) $(\overline{x} \vee y) \wedge (z \wedge y \vee x) \vee u$;

7) $(x \to y) \wedge (y \to z)$; 8) $(x \to y) \to (\overline{x} \wedge (y \vee z))$;

9) $((x \to y) \wedge (y \to z)) \to (x \to z)$; 10) $(x \to (y \to z)) \to (y \to x)$.

2. Построить РКС для $F(x, y, z)$, если известно, что:

1) $F(0,1,0) = F(1,0,1) = F(1,1,1) = 1$;

2) $F(1,0,1) = F(1,1,0) = 1$;

3) $F(0,0,1) = F(0,1,1) = F(1,0,1) = F(1,1,1) = 1$;

4) $F(1,1,0) = F(1,1,1) = 1$;

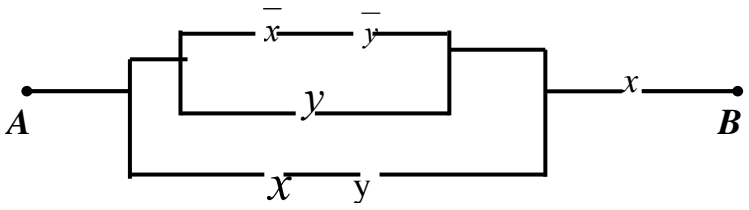
5) $F(0,0,1) = F(1,0,1) = F(1,0,0) = 1$;

6) $F(0,0,1) = F(0,1,0) = F(0,1,1) = F(1,0,1) = 1$,

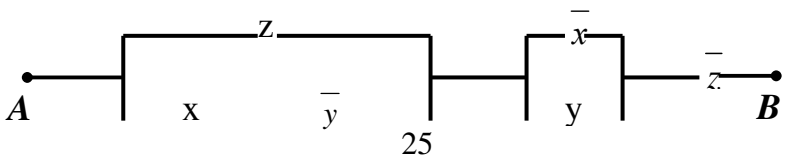
а остальные значения функции F равны нулю.

3. Упростить РКС:

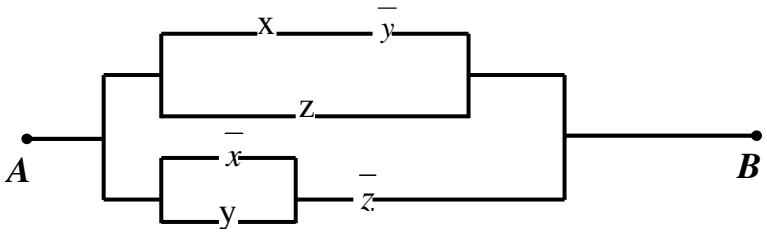
1)



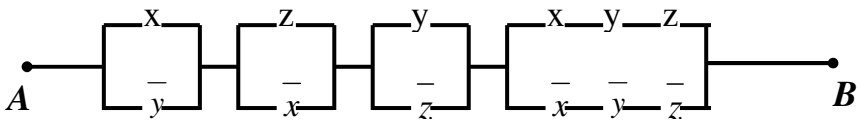
2)



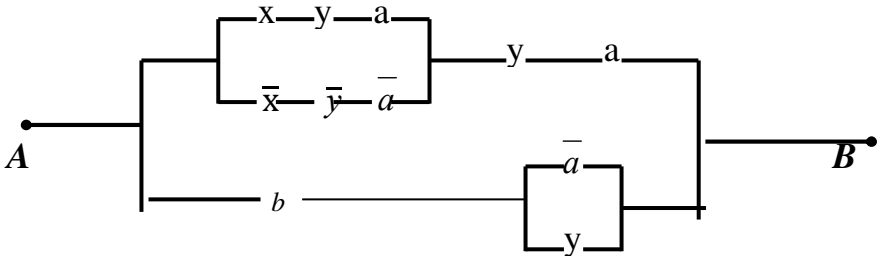
3)



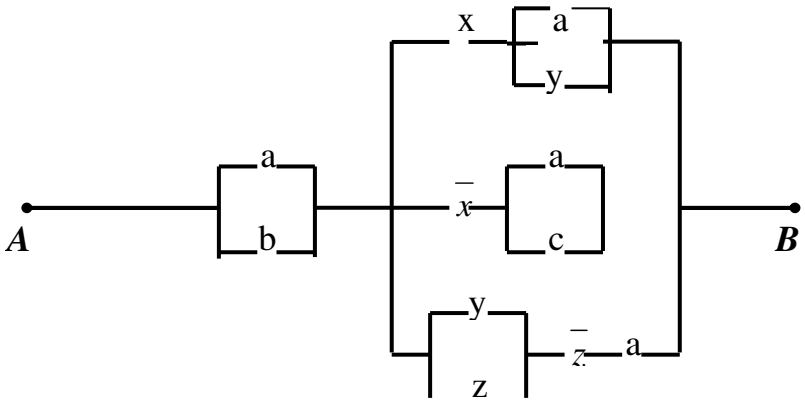
4)



5)

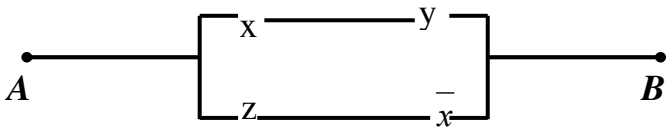


6)

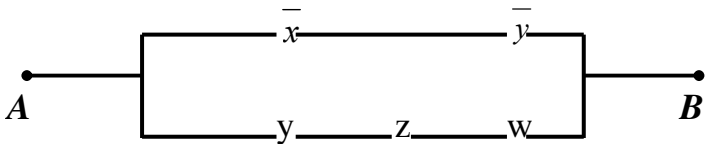


4. По данной схеме найдите функцию проводимости (СДНФ) и условия работы:

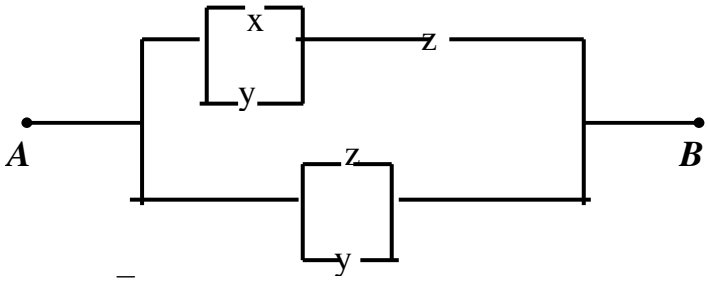
1)



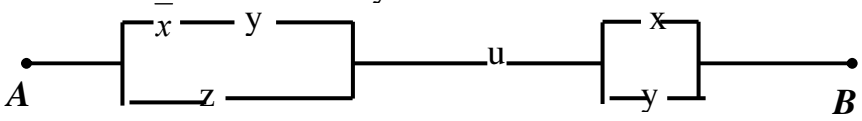
2)



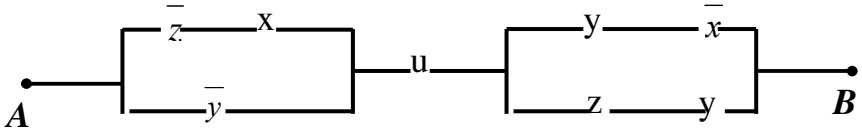
3)



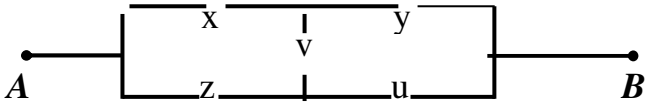
4)



5)

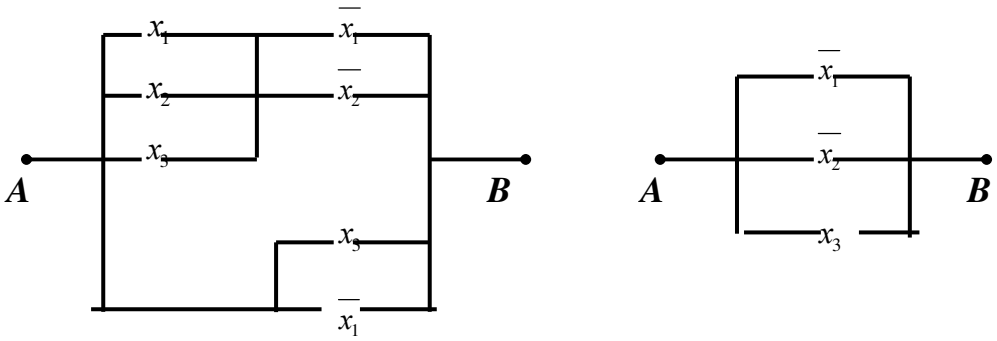


6)

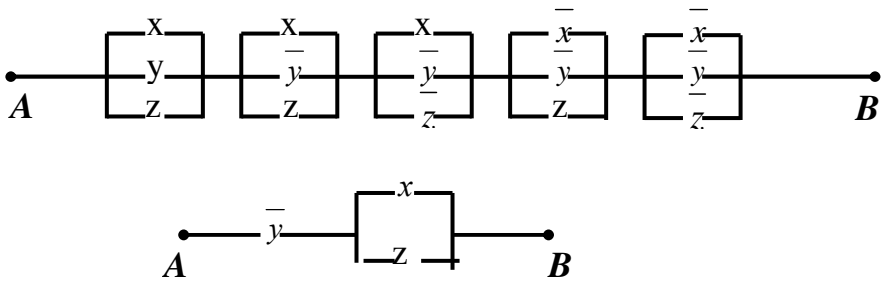


5. Проверьте равносильность схем:

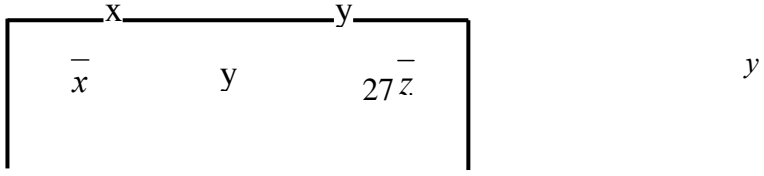
1)

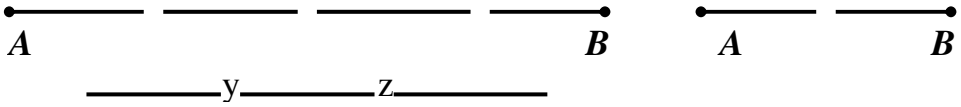


2)



3)




A
B
A
B
y
z

1.6. Суперпозиция функций. Замыкание набора функций. Замкнутые классы функций. Полные наборы. Базисы

Пусть имеется некоторый набор K, состоящий из конечного числа булевых функций. *Суперпозицией* функций из этого набора называется новая функция, полученная с помощью конечного числа применения двух операций: можно переименовать любую переменную, входящую в функцию из K; вместо любой переменной можно поставить функцию из набора K или уже образованную ранее суперпозицию.

Суперпозицию еще иначе называют сложной функцией.

Пример 1.17. Если дана одна функция $x|y$ (штрих Шеффера), то ее суперпозициями, в частности, будут следующие функции $x|x$, $x|(x|y)$, $x|(y|z)$.

Определение 1.10. *Замыканием* набора функций из K называется множество $[K]$ всех суперпозиций. Класс функций K называется *замкнутым*, если его замыкание совпадает с ним самим. Обозначение: $K = [K]$.

Свойства замыкания:

1. $K \subseteq [K]$
2. $\big[[K]\big] = [K]$
3. Если $K \subseteq G$, то $[K] \subseteq [G]$

Система функций называется *полной*, если ее замыкание совпадает со всем классом булевых функций $[K] = P_2$. Иначе говоря, полная система функций – это множество таких функций, через которые можно выразить все остальные булевы функции.

Неизбыточный полный набор функций называется *базисом* («неизбыточный» означает, что если какую-то функцию удалить из набора, то этот набор перестанет быть полным).

Примеры 1.18.

Примеры полных классов: а) $\Psi = P_2$;

б) $\Psi = \{\bar{x},\ x_1 \wedge x_2,\ x_1 \vee x_2\}$ (конъюнкция, дизъюнкция и отрицание являются полным набором, так как любая булева функция может быть

представлена в виде СКНФ или СДНФ, но не являются базисом, так как это набор избыточен, поскольку с помощью правил де Моргана можно удалить конъюнкцию или дизъюнкцию.

в) $\Psi = \{0,\ 1,\ x_1 \wedge x_2,\ x_1 \oplus x_2\}$ –любую булеву функцию можно представить в виде полинома Жегалкина (ясно, что функции конъюнкция, сложение по модулю 2 и константы 0 и 1 являются полным набором, но эти четыре функции также не являются базисом, поскольку 1⊕1=0, и поэтому константу 0 можно исключить из полного набора).

Легко видеть, что одним из способов проверки полноты какого-то набора K является проверка того, что через функции из этого набора выражаются функции другого полного набора (можно проверить, что через функции из K можно выразить конъюнкцию и отрицание или дизъюнкцию и отрицание).

Существуют такие функции, что одна такая функция сама является базисом (здесь достаточно проверить только полноту, неизбыточность очевидна). Такие функции называются шефферовскими функциями. Это название связано с тем, что штрих Шеффера является базисом. Напомним, что штрих Шеффера определяется следующей таблицей истинности:

$$\text{x}|y = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 0 \end{pmatrix} = \overline{x \wedge y}$$ («не и»), так как $x|x = \bar{x}$, т.е. отрицание является суперпозицией штриха Шеффера, а дизъюнкция $x \vee y = \overline{x|y} = (x|y)|(x|y)$, штрих Шеффера сам является базисом. Аналогично, стрелка Пирса является шефферовской функцией. Для функций 3-х или более переменных шефферовских функций очень много (конечно, выражение других булевых функций через шефферовскую функцию большого числа переменных сложно, поэтому в технике они редко используются).

Заметим, что вычислительное устройство чаще всего базируется на полном наборе функций (часто на базисах). Если в основе устройства лежат конъюнкция, дизъюнкция и отрицание, то для этих устройств важна

проблема минимизации ДНФ; если в основе устройства лежат другие функции, то полезно уметь алгоритмически минимизировать выражения через эти функции.

Перейдем теперь к выяснению полноты конкретных наборов функций. Для этого перечислим 5 важнейших классов функций:

- T_0 – это набор всех тех логических функций, которые на нулевом наборе принимают значение 0 (T_0 – это класс функций, *сохраняющих 0*).
- T_1 – это набор всех логических функций, которые на единичном наборе принимают значение 1 (T_1 – это класс функций, *сохраняющих единицу*)

Вектор значений функции $f \in T_0$ имеет вид $\tilde{\alpha}_f = (0,\ \alpha_1,\ \dots\ ,\ \alpha_{2^n-1})$, т.е. определено только значение на нулевом наборе переменных, свободных же $2^n - 1$. Следовательно, $|T_0| = 2^{2^n-1}$. Аналогично вычисляется количество функций класса T_1.

- L – класс *линейных* функций, т. е. функций, для которых полином Жегалкина содержит только первые степени переменных.

Различных линейных функций от переменных $x_1,\ \dots\ ,\ x_n$ столько же, сколько различных векторов $(\alpha_0,\ \alpha_1,\ \dots\ ,\ \alpha_n)$, т. е. 2^{n+1}.

Замечание. *Если* $n \geq 2$, *то линейная функция* в *таблице истинности может содержать только четное число единиц*. Действительно, если $f(x_1, x_2, \dots, x_n) = x_1 \oplus x_2 \oplus \dots \oplus x_n$, то легко видеть, что такая функция в таблице истинности содержит одинаковое число нулей и единиц, а именно $2^n/2$ единиц и нулей. Добавление фиктивной переменной приводит к увеличению числа единиц (и нулей) в два раза. Разумеется, нелинейная функция может иметь в таблице истинности как четное, так и нечетное число единиц.

- Класс S – класс *самодвойственных* функций. Функция n переменных называется самодвойственной, если на противоположных наборах она принимает противоположные значения, т.е. самодвойственная функция

$f(x_1, x_2, \dots, x_n)$ удовлетворяет условию $f(x_1, x_2, \dots, x_n) = \overline{f(\bar{x}_1, \bar{x}_2, \dots, \bar{x}_n)}$.

Функция $f \in S$ принимает противоположные значения на противоположных наборах переменных, поэтому для ее задания достаточно задать первую половину ее вектора значений $\tilde{\alpha}_f = (\underbrace{\alpha_0,\ \alpha_1, \dots\ ,\ \alpha_{2^{n-1}-1}}_{2^{n-1}},\ \overline{\alpha_{2^{n-1}-1}},\ \dots\ ,\ \overline{\alpha_0})$. Следовательно, количество самодвойственных функций, зависящих от n переменных, равно $|S| = 2^{2^n-1}$.

Замечание: таблица истинности самодвойственной функции не должна быть симметрична ни для одного набора значений переменных.

- M – класс *монотонных* функций. Опишем класс этих функций более подробно.

Пусть имеются 2 набора из n переменных: $s_1 = (x_1, x_2, \dots, x_n)$ и $s_2 = (y_1, y_2, \dots, y_n)$. Будем говорить, что набор s_1 меньше набора s_2, если все $x_i \le y_i$.

Очевидно, что не все наборы из n переменных сравнимы между собой (например, при n=2 наборы (0,1) и (1,0) не сравнимы между собой).

Функция от n переменных называется *монотонной*, если на меньшем наборе она принимает меньшее или равное значение. Разумеется, эти неравенства должны проверяться только на сравнимых наборах.

Пример 1.19. В нижеследующей таблице функции f_1, f_2 являются монотонными функциями, а функции f_3, f_4 – нет. Функции f_1, f_2 являются самодвойственными, а функции f_3, f_4 не являются.

x	y	f_1	f_2	f_3	f_4
0	0	0	0	0	1
0	1	1	0	1	0
1	0	0	1	1	0
1	1	1	1	0	1

Замечание. Естественный порядок переменных обеспечивает тот факт, что если какой-то набор меньше другого набора, то он обязательно

расположен в таблице истинности *выше* “большего” набора. Поэтому если в таблице истинности (при естественном порядке набора переменных) вверху стоят нули, а затем единицы, то эта функция точно является монотонной. Однако возможны инверсии, т.е. единица стоит до каких-то нулей, но функция является все равно монотонной. В этом случае наборы, соответствующие “верхней” единице и “нижнему” нулю должны быть *несравнимы*. Можно проверить, что функция, задаваемая таблицей истинности при естественном порядке набора переменных (00010101), является монотонной.

Теорема. *Классы функций* T_0, T_1, L, M, S *замкнуты*.

Это утверждение следует непосредственно из определения самих этих классов, а также из определения замкнутости.

Доказательство. а) Рассмотрим функцию $F = f(f_1, \dots, f_k)$, где $f, f_1, \dots, f_k \in T_0$. Покажем, что $F \in T_0$. Действительно,

$$F(0,\dots,0) = f(f_1(0,\dots,0),\dots,f_k(0,\dots,0)) = f(0,\dots,0) = 0\,.$$

Следовательно, класс T_0 замкнут.

б) Аналогично предыдущему доказывается замкнутость класса T_1.

в) Пусть $F = f(f_1, \dots, f_k)$, где $f, f_1, \dots, f_k -$ самодвойственные функции. Тогда $F^* = f^*(f_1^*,\dots,f_k^*) = f(f_1,\dots,f_k) = F$,

т. е. $F \in S$. Следовательно, класс S замкнут.

г) Пусть $F = f(f_1, \dots, f_k)$, где $f, f_1, \dots, f_k \in M$. Покажем, что $F \in M$.

Пусть $\tilde{x} = (x_1, \dots, x_n)$, $\tilde{x}_1 = (x_{11}, \dots, x_{1p_1})$, $\dots$, $\tilde{x}_k = (x_{k1}, \dots, x_{kp_k})$

Наборы переменных состоят из переменных, встречающихся у функций F, $f_1, \dots, f_k$ соответственно.

Возьмем два набора $\tilde{\alpha}^n$ и $\tilde{\beta}^n$ значений переменных $\tilde{x}$. Они определяют наборы $\tilde{\alpha}_1, \tilde{\beta}_1, \dots, \tilde{\alpha}_k, \tilde{\beta}_k$ значений переменных $\tilde{x}_1, \dots, \tilde{x}_k$, причем $\tilde{\alpha}_1 \preceq \tilde{\beta}_1, \dots, \tilde{\alpha}_k \preceq \tilde{\beta}_k$. Так как $f_1, \dots, f_k \in M$, то

$$f_1(\tilde{\alpha}_1) \le f_1(\tilde{\beta}_1),\dots,f_k(\tilde{\alpha}_k) \le f_k(\tilde{\beta}_k)\,.$$

Тогда $(f_1(\tilde{\alpha}_1), \dots, f_k(\tilde{\alpha}_k)) \preceq (f_1(\tilde{\beta}_1), \dots, f_k(\tilde{\beta}_k))$. Функция $f \in M$, поэтому

$f(f_1(\tilde{\alpha}_1),\ldots,f_k(\tilde{\alpha}_k)) \le f(f_1(\tilde{\beta}_1),\ldots,f_k(\tilde{\beta}_k))$.

Отсюда $F(\tilde{\alpha}) = f(f_1(\tilde{\alpha}_1),\ldots,f_k(\tilde{\alpha}_k)) \le f(f_1(\tilde{\beta}_1),\ldots,f_k(\tilde{\beta}_k)) = F(\tilde{\beta})$.

Следовательно, класс M замкнут.

д) Класс L замкнут, так как линейное выражение, составленное из линейных выражений, является линейным.

Класс K называется – ***предполным***, если K – не полный, но для любой функции $f \notin K$ класс $K_1 = K \cup \{f\}$ – полный.

Лемма. Классы Поста $T_0,\ T_1\ S,\ M,\ L$ попарно различны.

Доказательство. Для доказательства леммы приведем функции, лежащие в классах, но так, чтобы классы взаимно не поглощались. Рассмотрим функции $0,\ 1,\ \overline{x}$ и построим таблицу принадлежности классам. В таблице будем ставить «+», если функция принадлежит классу, и «–» в противном случае.

	T_0	T_1	S	M	L
0	+	–	–	+	+
1	–	+	–	+	+
$\overline{x}$	–	–	+	–	+

Если бы какие-нибудь два класса совпадали, то совпадали бы и соответствующие столбцы таблицы. Так как они не совпадают, делаем вывод о попарном различии классов.

В теории булевых функций очень большое значение имеет следующая теорема Поста.

Теорема Поста. *Для того*, *чтобы система функций K была полной, необходимо и достаточно, чтобы она целиком не содержалась ни в одном из пяти замкнутых классов* T_0, T_1, L, M, S.

Заметим, что необходимость этого утверждения очевидна, так как, если бы все функции из набора K входили в один из перечисленных классов, то и все суперпозиции, а значит, и замыкание набора входило бы в этот класс, и

класс *K* не мог быть полным. Достаточность этого утверждения доказывается довольно сложно, поэтому здесь не приводится.

Из этой теоремы следует довольно простой способ выяснения полноты некоторого набора функций. Для каждой из этих функций выясняется принадлежность к перечисленным выше классам. Результаты заносятся в так называемую *критериальную таблицу Поста*, где знак "+" означает принадлежность функции соответствующему классу, знак "–" означает, что функция в него не входит.

Для функций примера 1.19 построена таблица Поста:

f	T_0	T_1	L	M	S
f_1	+	+	+	+	+
f_2	+	+	+	+	+
f_3	+	–	+	–	–
f_4	–	+	+	–	–

В соответствии с теоремой Поста набор функций будет полным тогда и только тогда, когда в каждом столбце таблицы Поста имеется хотя бы один минус. Таким образом, из приведенной таблицы следует, что данные 4 функции не являются полной системой, как все функции линейные.

Пример 1.20.

Исследовать полноту системы функций:

$A=\{f_1 = xy \oplus z,\ f_2 = x \oplus y \oplus 1\}$

Решение:

Критериальная таблица Поста имеет вид:

f	T_0	T_1	L	M	S
f_1	+	–	–	–	–
f_2	–	+	+	–	–

Вывод: в каждом столбце имеется не менее одного минуса, значит система функций полная.

Полными наборами будут любые наборы, содержащие какой-либо базис.

Определение 1.11. Полная система булевых функций называется *базисом в* P_2, если никакая ее подсистема не является полной, т. е. 1) $[F] = P_2$, 2) для $\forall f \in F\ [F\backslash\{f\}] \neq P_2$.

Пример 1.21. Покажем, что полная система в примере 1.20 A=$\{f_1 = \text{xy}\oplus z,\ f_2 = \text{x}\oplus\text{y}\oplus 1\}$ является базисом, для этого достаточно показать, что после исключения любой функции из этой системы будет получена неполная система. Действительно, $A\backslash\{f_1\} \in T_1$, $A\backslash\{f_2\} \in T_0$, что позволяет сделать вывод о том, что система A – базис.

Непосредственно из таблицы Поста следует, что число базисных функций не может быть больше 5. Существует доказательство, что на самом деле это число меньше или равно 4.

Лемма. Из всякой полной системы можно выделить полную подсистему, содержащую не более четырех функций.

Доказательство. Так как $f_1 \notin T_0$, то либо $f_1 \notin S$ либо $f_1 \notin M$. Тогда полная система будет состоять из функций $\{f_1,\ f_2,\ f_4, f_5\}$ либо $\{f_1,\ f_2,\ f_3 f_5\}$.

Пример 1.22. **(нахождение базисов полной системы булевых функций).**

Из полной системы функций нужно выделить все базисы:

A=$\{f_1 = \text{x}\oplus y, f_2 = \text{xy}\oplus z,\ f_3 = \text{x}\oplus\text{y}\oplus z \oplus 1\,, f_3 = \text{x}y\oplus\text{y}z\oplus zx\}$

Критериальная таблица Поста имеет вид:

f	T_0	T_1	L	M	S
f_1	+	–	+	–	–
f_2	+	–	–	–	–
f_3	–	–	+	+	–
f_4	+	+	–	+	+

По таблице составим КНФ, в которой элементарные дизъюнкции соответствуют столбцам таблицы и включают в качестве слагаемых символы тех функций, которые не входят в класс, соответствующий столбцу.

В данном примере имеем: $K = f_3(f_1 \vee f_2 \vee f_3)(f_2 \vee f_4)(f_1 \vee f_2)(f_1 \vee f_2 \vee f_3)$.

Перемножая скобки и используя для упрощения равенства вида $AA = A$,

$A \cdot (A \vee B) = A$, $A \vee AB = A$, приведем КНФ к ДНФ, в которой упрощения $A \vee AB = A$ больше невозможны.

В данном примере имеем:

$K = f_3(f_2 \vee f_4)(f_1 \vee f_2) = f_3 f_2 f_1 \vee f_3 f_4 f_1 \vee f_3 f_2 \vee f_3 f_4 f_2 = f_3 f_4 f_1 \vee f_3 f_2$

Из полученной ДНФ выпишем подмножества функций, соответствующие слагаемым в этой ДНФ. Это и будут искомые базисы.

В данном примере получили два базиса: $\mathrm{B}_1 = \{f_1, f_3, f_4\}$ $\mathrm{B}_2 = \{f_3, f_2\}$

Задачи

1. Назовем операциями Шеффера следующие операции:

$A|B = \overline{AB} = \overline{A \vee B}$; $A \downarrow B = \overline{AB} = \overline{A} \vee \overline{B}$.

Выразить через них отрицание, дизъюнкцию и конъюнкцию.

2. Перечислить все самодвойственные функции от двух переменных.

3. Найти пары двойственных функций и все самодвойственные функции в множестве:

а) $f_1 = x \wedge y$, $f_2 = x \vee y$, $f_3 = x \to y$; $f_4 = x \leftrightarrow y$; $f_5 = x \oplus y$, $f_6 = x \downarrow y$, $f_7 = x|y$;

б) $f_1 = x \to y$, $f_2 = (\bar{x} \to \bar{y}) \to (y \to x)$, $f_3 = x \oplus y \oplus z$, $f_4 = xy \oplus xz \oplus yz$, $f_5 = xy \vee xz \vee yz$,

$f_6 = (x \to y) \cdot (\overline{y} \to \overline{x}), f_7 = \overline{x} \cdot y$.

Ответ: а) пары двойственных: f_1 и f_2, f_4 и f_5, f_6 и f_7; самодвойственных функций нет; б) $f_1 = f_6 = f_7^*, f_3^* = f_3, f_4^* = f_4, f_5^* = f_5$.

4. Пользуясь принципом двойственности доказать самодвойственность функций:

a) $f = xy \vee yz \vee zx$

b) $f = (x \vee \overline{y} \vee z)t \vee x \cdot \overline{y} \cdot z$

c) $f(x_1, x_2, x_3) = x_2 \downarrow (\bar{x}_3 \leftrightarrow \bar{x}_1) \wedge x_2$

d) $f = x \oplus y \oplus z \oplus 1$

5. Выяснить, является ли самодвойственной функция f, заданная векторно:

а) $\tilde{\alpha}_f = (1010)$; б) $\tilde{\alpha}_f = (10010110)$;

в) $\tilde{\alpha}_f = (10110101)$;

г) $\tilde{\alpha}_f = (1100100101101100)$.

6. Перечислить все монотонные функции от двух переменных.

7. Выяснить, является ли монотонной функция f:

a) $f = (x \oplus y) \cdot (x \sim y)$;

b) $f = xy \oplus xz \oplus zy$;

c) $f = x_1 \to (x_2 \to x_3)$;

d) $f = x_1 \oplus (x_2 \to x_3) \oplus x_4$.

8. Выяснить, является ли монотонной функция f, заданная векторно:

а) $\tilde{\alpha}_f = (0110)$; б) $\tilde{\alpha}_f = (10110111)$;

в) $\tilde{\alpha}_f = (00010111)$; г) $\tilde{\alpha}_f = (001000110111111)$.

9. Проверьте свойства булевой функции (линейность, самодвойственность, монотонность) и построить соответствующую РКС:

a) $f(x, y, z) = (\mathrm{x} \oplus 1) \wedge (\mathrm{y} \oplus 1) \wedge \overline{z} \vee y \wedge z$

b) $f(x, y, z) = \overline{\mathrm{x}} \wedge (\overline{y} \wedge z \vee y \wedge \overline{z}) \vee (y \oplus z \oplus 1) \wedge x$.

10. Выясните, полны ли системы функций:

a) $A = \{\to\}$

b) $A = \{\downarrow\}$

c) $A = \{x \to y\ , \bar{x}\}$

d) $A=\{x \to y, \bar{x}, x \wedge y, x \vee y\}$

e) $A=\{x \leftrightarrow y, \bar{x}, x \vee y\}$

f) $A=\{x \oplus y, \bar{x}, \}$

g) $A=\{x \oplus y, \bar{x}, x \wedge y \}$

h) $A=\{xy,\ x \vee y,\ xy \vee yz \vee zx\}$

i) $A=\{xy,\ x \vee y,\ x \oplus y \oplus z \oplus 1\}$;

j) $A=\{1,\ \bar{x},\ x(y \leftrightarrow z) \oplus \bar{x}(y \oplus z),\ x \leftrightarrow y\}$;

k) $A=\{0,\ \bar{x},\ x(y \oplus z) \oplus yz\}$;

l) $A=\{\ \bar{x},\ x(y \leftrightarrow z) \leftrightarrow (y \vee z),\ x \oplus y \oplus z\}$;

m) $A=\{\ \bar{x},\ x(y \leftrightarrow z) \leftrightarrow yz,\ x \oplus y \oplus z\}$;

n) $A=\{xy(x \oplus y),\ xy \oplus x \oplus y,\ 1,\ xy \oplus yz \oplus zx)\}$;

o) $A=\{(y \leftrightarrow x) \oplus z,\ x \wedge y \oplus z,\ 0\}$;

p) $A=\{\ x \to y,\ \bar{x} \to \bar{y}x,\ x \oplus y \oplus z,\ 1\}$.

q) $A=\{0,\ 1,\ x \oplus y \oplus z,\ xy \oplus yz \oplus zx,\ xy \oplus z,\ x \vee y\}$.

11. Выясните, полны ли системы функций:

a) $B=\{f_1 = (10),\ f_2 = (00110111)\}$

b) $B=\{f_1 = (0110),\ f_2 = (1100\ 0011),\ f_3 = (1001\ 0110)\}$

c) $B=\{f_1 = (0111),\ f_2 = (0101\ 1010),\ f_3 = (0111\ 1110)\}$

d) $B=\{f_1 = (0111),\ f_2 = (1001\ 0110)\}$

e) $B=\{f_1 = (0101),\ f_2 = (1110\ 1000),\ f_3 = (0110\ 1001)\}$

f) $B=\{f_1 = (1001),\ f_2 = (1110\ 1000)\}$

g) $B=\{f_1 = (11),\ f_2 = (0111),\ f_3 = (00110\ 111)\}$

h) $B=\{f_1 = (11),\ f_2 = (00),\ f_3 = (00110\ 101)\}$

12. Выделите всевозможные базисы из полной в P_2 системы:

a) $C=\{1,\ \bar{x},\ xy(x \oplus y),\ x \oplus y \oplus xy \oplus yz \oplus zx\}$

b) $C=\{0,\ x \oplus y,\ x \to y,\ xy \leftrightarrow zx\}$

c) $C=\{0,\ 1,\ x \oplus y \oplus z,\ xy \oplus yz \oplus zx,\ xy \oplus z,\ x \vee y\}$

d) $C=\{xy,\ x \vee y,\ xy \vee z,\ x \oplus y,\ x \to y\}$

e) $C=\{xy \oplus z,\ x \oplus y \oplus 1,\ x\bar{y},\ \bar{x}\}$

f) $C=\{xy \vee \bar{z},\ \bar{x},\ x \to y,\ 0,\ x \oplus zy\}$

g) $C=\{xy,\ xy \vee z,\ \bar{x},\ x \oplus y,\ x \to y,\ \bar{x}\}$

h) $C=\{x \oplus y,\ x \leftrightarrow y,\ 0,\ x \oplus y \oplus z,\ xy,\ x \to y\}$

Раздел 2. Частично - рекурсивные функции

2.1. Операция суперпозиции и примитивной рекурсии

Функцию, значения которой могут находиться с помощью некоторого алгоритма, называют *вычислимой* (или *эффективно вычислимой*) функцией.

Определение 2.1. Функцию $f\colon X \to Y$ будем называть *частичной*, если она определена не для каждого значения $x \in X$. Множество тех $x \in X$, для которых однозначно указано соответствующее значение функции f, называется *областью определения* функции. Если область определения D_f функции f совпадает со всем множеством X (т. е. $D_f = X$), то функция называется всюду определенной.

Будем рассматривать только функции $y = f\ (x_1,\ x_2, \dots,\ x_n)$, где $y,\ x_i \in N_0$ – целые неотрицательные числа. При этом функции могут быть как всюду определенными, так и не всюду определенными, т. е. частичными.

Опишем класс числовых функций, совпадающий с множеством всех вычислимых функций. Этот класс строится индуктивно.

В качестве базисных выбираются простейшие функции:

1) $o(x) = 0$ – *нулевая функция* или *оператор аннулирования*;

2) $s(x) = x + 1$ – *функция следования* или *оператор сдвига*;

3) $I_m^n(x_1,\ x_2, \dots,\ x_n) = x_m$ – *функции выбора аргумента* или *операторы проектирования* (здесь $1 \le m \le n$).

Ясно, что все три функции всюду определены и вычислимы (их правильную вычислимость можно установить с помощью машины Тьюринга).

Из исходных базисных функций будем образовывать другие с помощью трех операций: суперпозиции, примитивной рекурсии и минимизации.

Определение 2.2. (операция суперпозиции). Говорят, что n -местная функция $F\ (x_1,\ x_2, \dots,\ x_n)$ получена из m -местной функции ϕ и n -местных

функций $f_1, \dots, f_m$ с помощью *операции суперпозиции*, если для всех $(x_1,\ x_2, \dots,\ x_n)$ справедливо равенство: $F\ (x_1,\ x_2, \dots, x_n) =$

$= \phi\ (f_1(x_1, \dots, x_n), f_2(x_1, \dots, x_n), \dots f_m(x_1, \dots, x_n))$.

Операция суперпозиции обозначается через $F = S(\phi,\ f_1, \dots,\ f_m)$.

При этом функция F определена на наборе $\tilde{\alpha}^n = (a_1,\ a_2, \dots,\ a_n)$ тогда и только тогда, когда каждая функция $f_i\ (i = 1, \dots, m)$ определена на наборе $\tilde{\alpha}^n$ и, кроме того, функция ϕ определена на наборе $(f_1(\tilde{\alpha}^n), \dots, f_m(\tilde{\alpha}^n))$; в этом случае $F(\tilde{\alpha}^n) = \phi\ (f_1(\tilde{\alpha}^n), f_2(\tilde{\alpha}^n), \dots, f_m(\tilde{\alpha}^n))$.

Пример 2.1. Постоянные функции $f(x) = 1$ и $f(x) = 3$ получаются суперпозицией из базисных функций $o(x)$ и $s(x)$ следующим образом: $f(x) = 1 = s\ (o(x))$ и $f(x) = 3 = s(s(s(o(x))))$. В самом деле, чтобы из любого числа x получить фиксированное число, нужно сначала числу x сопоставить число 0: $o(x) = 0$, а затем к полученному результату, т. е. к нулю, с помощью функции следования $s(x)$ прибавлять по единице 1 до тех пор, пока не получится нужное число.

Функция $O(x_1, x_2, \dots, x_n) = 0$ получается с помощью операции суперпозиции (подстановки) из функций $o(x)$ и $I_1^n(x_1,\ x_2, \dots,\ x_n)$ следующим образом: $O(x_1,\ x_2, \dots,\ x_n) = o(\ I_1^n(x_1,\ x_2, \dots,\ x_n))$.

Функции $g(x) = x + 3$ и $S_m^n(x_1,\ x_2, \dots,\ x_n) = x_m + 1$ получаются с помощью подстановки из функций $s(x)$ и

$I_m^n(x_1,\ x_2, \dots,\ x_n)$:

$g(x) = x + 3 = s(s(s(x)))$;

$S_m^n(x_1,\ x_2, \dots,\ x_n) = x_m + 1 = s(I_m^n(x_1,\ x_2, \dots,\ x_n))$.

Определение 2.3 (операция примитивной рекурсии). Говорят, что функция $f\ (x_1,\ x_2, \dots,\ x_n)$ от n переменных получается с помощью *операции примитивной рекурсии* из функций $g(x_1,\ x_2, \dots,\ x_{n-1})$ от $n-1$ переменных и $h\ (x_1, \dots,\ x_n,\ x_{n+1})$ от $n+1$ переменных [обозначение $f = R(g, h)$], если для любых $x_1,\ x_2, \dots,\ x_n$ выполняются равенства:

1)$f(x_1,\ x_2, \dots,\ x_{n-1},\ 0) = g(x_1, \dots,\ x_{n-1})$; (1)

2)$f(x_1,\ x_2, \dots,\ x_{n-1}, k+1) = h(x_1, \dots,\ x_{n-1},\ k, f(x_1, \dots,\ x_n,\ k)), k \in N_0$.

Пара равенств (1) называется схемой примитивной рекурсии для функции f по переменной x_n и задает примитивно рекурсивное описание функции f с помощью функций g и h.

Рекурсия в этом определении ведется по последней переменной функции f и выражает индуктивный способ определения: зная значение функции при $k = 0$, мы можем вычислить значение функции для $k = 1$, затем для $k = 2$ и т. д.

Замечание: схему примитивной рекурсии можно записывать по любой переменной, указывая отдельно, по какой переменной ведется рекурсия.

При задании примитивно рекурсивного описания функции $f(x)$, зависящей от одной переменной, схема примитивной рекурсии имеет вид:

1) $f(0) = a$;

2) $f(y+1) = h(y, f(y)), y \geq 0$,

где a – постоянная одноместная функция, равная числу a, т. е. константа (число из множества N_0 – натурального ряда с нулем).

Пример 2.2. Покажем, что следующие функции могут быть получены из простейших с помощью оператора примитивной рекурсии:

а)$sum\ (x,\ y) = x + y$;

б)$p(x,\ y) = x \cdot y$;

в)$x \dot{-} y = \begin{cases} x - y, & если\ x \geq y \\ 0, & если\ x < y \end{cases}$.

Решение:

а) Для функции $sum\ (x,\ y) = x + y$ верны следующие равенства:

1) $sum\ (x, 0) = x + 0 = x = I_1^1(x)$;

2) $sum\ (x, y+1) = x + (y+1) = (x+y) + 1 = sum\ (x,\ y) + 1 = s(sum\ (x, y))$,

это и есть схема примитивной рекурсии для функции $sum\ (x,\ y) = x + y$, основывающаяся на базисных функциях $I_1^1(x)$ и $s(x)$.

б) Аналогично для операции умножения запишем соотношения (1):

1) $p(x,\ 0) = x \cdot 0 = 0 = o(x)$;

2) $p(x,\ y+1) = x \cdot (y+1) = x \cdot y + x =$

$$= p(x,\ y) + x = sum\ (x,\ p(x,\ y)).$$

Записанные выражения показывают, что функция $p(x,\ y) = x \cdot y$ получается из базисной функции $o(x)$ и функции $sum\ (x,\ y) = x + y$ с помощью схемы примитивной рекурсии и суперпозиции.

в) Во-первых, отметим, что функция $x \dot{-} 1$, получающаяся из функции $x \dot{-} y$ фиксированием второго аргумента, удовлетворяет следующим соотношениям:

1) $0 \dot{-} 1 = 0 = o(x)$;

2) $(x+1) \dot{-} 1 = x = I_1^2(x,\ y)$,

т. е. функция $x \dot{-} 1$ получена из простейших функций $o(x)$ и $I_1^2(x,\ y)$ с помощью оператора примитивной рекурсии.

Во-вторых, исходя из определения усеченной разности, получаем, что функция $x \dot{-} y$ для любых x и y удовлетворяет следующим равенствам:

1) $x \dot{-} 0 = x = I_1^1(x) = g(x)$;

2) $x \dot{-} (y+1) = (x \dot{-} y) \dot{-} 1 = z \dot{-} 1 = h\ (x, y, z)$.

Эти соотношения показывают, что двухместная функция $x \dot{-} y$ получена с помощью операции примитивной рекурсии из одноместной простейшей функции $g(x) = I_1^1(x)$ и трехместной функции $h(x, y, z) = z \dot{-} 1$, т. е. $x \dot{-} y = R\ (I_1^1(x),\ z \dot{-} 1)$.

Определение 2.4. Функция называется *примитивно рекурсивной*, если она может быть получена из базисных функций с помощью конечного числа применений операций суперпозиции и примитивной рекурсии.

Замечание. Все основные функции арифметики, алгебры и анализа (с поправкой на целочисленность) являются примитивно рекурсивными. Так, например, можно показать, что примитивно рекурсивными функциями являются $x + y$, $x \cdot y$, x^n, x^y и, следовательно, все многочлены с натуральными коэффициентами.

Так как исходные базисные функции являются всюду определенными и операции подстановки и примитивной рекурсии сохраняют всюду определенность, то из определения примитивно рекурсивной функции следует, что каждая примитивно рекурсивная функция является всюду определенной.

Пример 2.3. Докажем, что функция $q(x,y)=\left[\frac{x}{y}\right]$ – целая часть от деления x на y (здесь полагаем $\left[\frac{x}{0}\right]=x$) является примитивно рекурсивной.

Решение: выразим данную функцию через суперпозицию известных примитивно рекурсивных функций. Согласно определению функции $q(x,y)=\left[\frac{x}{y}\right]$, при $y>0$ число $\left[\frac{x}{y}\right]=n$ удовлетворяет неравенствам: $ny\le x\le(n+1)y$. Отсюда видно, что n равно числу нулей в последовательности $1y\dot{-}x$, $2y\dot{-}x$,…, $ny\dot{-}x$,…, $xy\dot{-}x$. Поэтому для $y>0$ имеем формулу $\left[\frac{x}{y}\right]=\sum_{i=1}^{x}\overline{sg}(iy\dot{-}x)$. Непосредственная проверка показывает, что эта формула верна и при $y=0$. Ввиду примитивной рекурсивности всех функций, участвующих в этой суперпозиции, примитивно рекурсивной будет и результирующая функция $\left[\frac{x}{y}\right]$ как их суперпозиция.

Пример 2.4. Обосновать примитивную рекурсивность функции $f(x,y)=x+(2\dot{-}y)$.

Решение: запишем схему примитивной рекурсии для функции $f(x,y)$, ведя рекурсию по переменной x:

1) $f(0,y)=2\dot{-}y=g(y)$;

2) $f(x+1,\ y)=x+1+(2\dot{-}y)=s(f(x,\ y))=$

$=I_3^3(x,y,s(f(x,\ y)))$.

Из этой схемы следует, что для описания функции $f(x,y)$ достаточно иметь функцию $g(y)=2\dot{-}y$ и функцию $h\ (x,\ y,\ z)=I_3^3(x,\ y,\ s(z))$. Очевидно, что функция $h\ (x,\ y,\ z)$ представима в виде суперпозиции простейших функций.

Дадим примитивно рекурсивное описание функции $g(y)$.

Имеем $g(0) = 2\dot{-}0 = 2 = s(s(0))$ и $g(y+1) = h_1(y,\ g(y)) = 2\dot{-}(y+1) = 1\dot{-}y$.

Значит, $h_1(y,\ z) = I_1^2(\phi(y),\ z)$, где $\phi(y) = 1\dot{-}y$. Следовательно, надо еще построить примитивно рекурсивное описание функции $\phi(y) = 1\dot{-}y$, которая, как нетрудно заметить, есть $\overline{sg}y$.

Легко увидеть, что $\phi(0) = 1 = s(0)$ и $\phi(y+1) = 0$.

Итак, $f(x,\ y)$ строится из простейших функций, с помощью операций суперпозиции и примитивной рекурсии. Значит, она примитивно рекурсивная функция.

Пример 2.5. Применяя операцию примитивной рекурсии к функциям $g(x)$ и $h(x,\ y,\ z)$ по переменной y, построить функцию $f(x,y) = R\ (g,\ h)$, записав ее в «аналитической» форме:

1) $g(x) = x, h\ (x,\ y,\ z) = x + z$;

2) $g(x) = x, h\ (x,\ y,\ z) = x + y$;

3) $g(x) = x, h\ (x,\ y,\ z) = z^x$;

4) $g(x) = x, h\ (x,\ y,\ z) = x^z$.

Решение: 1) По заданным функциям $g(x)$ и $h\ (x,\ y,\ z)$ запишем схему примитивной рекурсии для искомой функции $f(x,\ y)$ по переменной y:

$$\begin{cases} f(x,\ 0) = x \\ f(x,\ y+1) = x + f(x,\ y) \end{cases}.$$

Последовательно находя значения $f(x,\ y)$ для $y = 0,\ 1,\ 2, \ldots$, построим следующую таблицу:

y	0	1	2	3	…	n	…
$f(x,\ y)$	x	$x + x$	$x + 2x$	$x + 3x$	…	$x + nx$	…

Используя табличное задание $f(x,\ y)$, можно записать следующее аналитическое выражение искомой функции: $f(x,\ y) = x + y \cdot x$ или $f(x,\ y) = x \cdot (y+1)$.

2) Используя схему примитивной рекурсии $\begin{cases} f(x,\ 0) = x \\ f(x,\ y+1) = x + y \end{cases}$, составим таблицу значений $f(x,\ y)$:

y	0	1	2	3	…	n	…
$f(x,\ y)$	x	x	$x+1$	$x+2$	…	$x+n-1$	…

Используя табличное задание $f(x,\ y)$, получаем, что $f(x,y) = \begin{cases} x, & \text{если} y = 0 \\ x + y - 1, & \text{если} y \geq 1 \end{cases}$, откуда следует, что можно записать следующее аналитическое выражение искомой функции: $f(x,\ y) = x + y \dot{-} 1$.

3) Используя схему примитивной рекурсии:

$\begin{cases} f(x,\ 0) = x \\ f(x,\ y+1) = f(x,\ y)^x \end{cases}$, найдем значения функции $f(x,\ y)$ для $y = 0,\ 1,\ 2, \ldots$:

$f(x,\ 1) = f(x,\ 0)^x = x^x$;

$f(x,\ 2) = f(x,\ 1)^x = (x^x)^x = x^{x^2}$;

$f(x,\ 3) = f(x,\ 2)^x = \left(x^{x^2}\right)^x = x^{x^3}$;

$f(x,\ 4) = f(x,\ 3)^x = \left(x^{x^3}\right)^x = x^{x^4}$ … и т. д.

Зададим функцию таблично:

y	0	1	2	3	…	n	…
$f(x,\ y)$	x	x^x	x^{x^2}	x^{x^3}	…	x^{x^n}	…

Аналитическое выражение искомой функции: $f(x,\ y) = x^{x^y}$.

4) Используя схему примитивной рекурсии:

$\begin{cases} f(x,\ 0) = x \\ f(x,\ y+1) = x^{f(x,\ y)} \end{cases}$, найдем значения функции $f(x,\ y)$ для $y = 0,\ 1,\ 2, \ldots$: $f(x,\ 1) = x^{f(x,\ 0)} = x^x$; $f(x,\ 2) = x^{f(x,\ 1)} = x^{x^x}$; $f(x,\ 3) = x^{f(x,\ 2)} = x^{x^{x^x}}$; $f(x,\ 4) = x^{f(x,\ 3)} = x^{x^{x^{x^x}}}$ и т. д.

Зададим функцию таблично:

y	0	1	2	3	…	n	…
$f(x,\ y)$	x	x^x	x^{x^x}	$x^{x^{x^x}}$	…	$x^{x^{\cdot^{\cdot^{\cdot^x}}}}$	…

Аналитическое выражение искомой функции:

$f(x,\ y) = x^{x^{\cdot^{\cdot^{\cdot^x}}}\}y раз}$.

2.6. Операция минимизации (*μ-оператор)*

Определение 3.5 (операция минимизации). Говорят, что функция $g(x_1,\ x_2, \dots,\ x_n)$ получена из функции $f(x_1,\ x_2, \dots,\ x_n)$ с помощью *операции минимизации или с помощью μ-оператора* (обозначение $g = M(f)$ или $g(x_1,\ x_2, \dots,\ x_n) = \mu_z(f(x_1,\ x_2, \dots,\ x_{n-1},\ z) = x_n)$), если выполнено условие $g(x_1,\ x_2, \dots,\ x_n)$ определена и равна z тогда и только тогда, когда $f(x_1, \cdots x_2, \dots,\ x_{n-1},\ 0)$, $f(x_1,\ x_2, \dots,\ x_{n-1},\ 1)$,…, $f(x_1,\ x_2, \dots,\ x_{n-1},\ z-1)$ определены и не равны x_n, а $f(x_1,\ x_2, \dots,\ x_{n-1},\ z) = x_n$.

Замечание 1. Операцию минимизации можно применять по любой переменной, входящей в функцию $f(x_1,\ x_2, \dots,\ x_n)$, но всегда нужно указывать, по какой переменной эта операция проводится.

Замечание 2.

Вместо функции $g(x_1,\ x_2, \dots,\ x_n) = \mu_z(f(x_1,\ x_2, \dots,\ x_{n-1},\ z) = x_n)$ можно рассмотреть функцию $g(x_1,\ x_2, \dots,\ x_n) = \mu_z(f(x_1,\ x_2, \dots,\ x_{n-1},\ x_n) = 0)$.

Определяемый класс вычислимых функций при этом будет тот же самый.

Оператор минимизации является удобным средством для построения обратных функций. Так, функция $f^{-1}(x) = \mu_y(f(y) = x)$ (наименьший y такой, что $f(y) = x$) является обратной для функции $f(x)$. Поэтому в применении к одноместным функциям оператор минимизации иногда называют оператором обращения.

Определение 2.6. Функция называется *частично рекурсивной*, если она может быть получена из базисных функций $o(x)$, $s(x)$, I_m^n с помощью применения конечного числа раз операций суперпозиции, примитивной рекурсии и минимизации. Всюду определенная частично рекурсивная функция называется *общерекурсивной*.

***Тезис Черча*:** всякая эффективно вычислимая функция является частично рекурсивной.

Принятие данного тезиса позволяет истолковать доказательство, что некоторая функция не является частично рекурсивной, как доказательство отсутствия алгоритма вычисления ее значений.

Пример 2.6. Применить операцию минимизации к следующим функциям:

а) $f(x) = 5;$ б) $f(x) = x + 5;$ в) $f(x) = x - 5;$

$f(x) = x + 5;$ г)$f(x) = \begin{cases} 3x + 2, & если\, x \neq 2 \\ не\ определено, & если\, x = 2 \end{cases}.$

Результирующую функцию представить в «аналитической» форме.

Решение: а) Запишем функцию $g(x)$, полученную применением μ-оператора к функции $f(x) = 5$:

$g(x) = \mu_y(f(y) = x) = \mu_y(5 = x)$. Найдем значения функции $g(x)$ для $x = 0,\ 1,\ 2, \ldots$ и оформим их в виде таблицы:

x	0	1	2	3	4	5	6	7	8	...
$g(x)$	–	–	–	–	–	0	–	–	–	...

Получили, что функция $g(x)$ определена только в одной точке при x $=$ 5, значит, аналитическое выражение для $g(x)$ может иметь вид:

$g(x) = \overline{sg}|x - 5| - 1.$

б) Применим μ-оператор к функции $f(x) = x + 5$ и получим новую функцию $g(x) = \mu_y(f(y) = x) = \mu_y(y + 5 = x)$.

Таблица значений для $g(x)$:

x	0	1	2	3	4	5	6	7	8	9	...
$g(x)$	–	–	–	–	–	0	1	2	3	4	...

Аналитическое выражение для $g(x)$ может иметь вид: $g(x) = x - 5$.

в) Применим μ-оператор к функции $f(x) = x - 5$ и получим новую функцию $\phi\ (x) = \mu_y(f(y) = x) = \mu_y(y - 5 = x)$.

Таблица значений для $\phi\ (x)$:

x	0	1	2	3	4	5	6	7	8	9	...
$\phi\ (x)$	–	–	–	–	–	–	–	–	–	–	...

Получили, что функция $\phi\ (x)$ нигде не определена, так как для любого x левая часть уравнения $y - 5 = x$ не определена для малых значений y. Например, при $x = 6$ получим уравнение $y - 5 = 6$, которое имеет решение $y_0 = 11$. Но в определении оператора минимизации есть еще одно требование: левая часть уравнения $y - 5 = 6$ должна быть определена для всех $y \leq y_0$.

В этой задаче существуют $y < 11$, при которых значение выражения $y - 5$ не является целым неотрицательным числом, а значит, левая часть уравнения $y - 5 = 6$ не определена, например, при $y = 0$ получаем $0 - 5 = -5 \notin N_0$.

Аналитическое выражение для $\phi\ (x)$ может иметь, например, вид:

$g(x) = -x - 1$ или $g(x) = -\frac{1}{x}$, и т. д. (любая формула, не принимающая значения из N_0, является аналитическим выражением для $\phi\ (x)$).

г) Применим операцию минимизации к функции

$$f(x) = \begin{cases} 3x + 2, & \text{если } x \neq 2 \\ \text{не определено, если } x = 2 \end{cases}.$$

Для каждого $x_0 \in N_0$ ищем минимальное решение уравнения $f(y) = x_0$.

Так как множеством значений функции $f(x)$ является множество $\{3n + 2 |\ n \neq 2\} = \{2,\ 5\} \cup \{11,\ 14,\ 17, \ldots,\ 3n + 2, \ldots\}$, то уравнение $f(y) = x_0$ имеет решения лишь при $x_0 = 2,\ 5,\ 11,\ 14, \ldots$; для всякого такого x_0 решение единственное (оно равно $\frac{x_0 - 2}{3}$).

Принимая во внимание, что функция $f(x)$ при $x = 2$ не определена, заключаем, что найденные решения $\frac{x_0 - 2}{3}$, превосходящие 2, т. е. 3, 4, …, не являются допустимыми (см. определение оператора минимизации).

Итак, функция $g(x) = \mu_x f(x)$ определена только при $x = 2$ и $x = 5$; $g(2) = 0, g(5) = 1$.

В качестве «аналитической» записи функции $g(x)$ можно взять формулу $f(x) = \frac{x-2}{3} + \overline{sg}(6-x)$, так как функция $\frac{x-2}{3}$ определена только при $x = 2,\ 5,\ 8,\ 11, \ldots,\ 3n+2, \ldots$, а функция $\overline{sg}(6-x)$ – только для $x \leq 6$, причем $\overline{sg}(6-2) = \overline{sg}(6-5) = 0$.

Пример 2.7. (действие оператора минимизации для получения обратных функций).

а) Пусть дана функция $f(x,y) = x + y$. Применим оператор минимизации по переменной x: $d(x,y) = \mu_z(y + z = x) =$

$= \mu_z(sum\ (I_2^3(x,\ y,\ z),\ I_3^3(x,y,z) = I_1^3(x,\ y,\ z))$.

Вычислим, например, $d(7,\ 2)$. Для этого нужно положить x $= 7$, $y = 2$ и, придавая переменной z последовательно значения 0, 1, 2, ..., каждый раз вычислять сумму $y + z$. Как только она станет равной 7, то соответствующее значение z принять за значение $d(7,\ 2)$.

Вычисляем: $z = 0,\quad 2 + 0 = 2 \neq 7;$

$$z = 1,\quad 2 + 1 = 3 \neq 7;$$

$$z = 2,\quad 2 + 2 = 4 \neq 7;$$

$$z = 3,\quad 2 + 3 = 5 \neq 7;$$

$$z = 4,\quad 2 + 4 = 6 \neq 7;$$

$$z = 5,\quad 2 + 5 = 7.$$

Таким образом, $d(7,\ 2) = 5$.

Попробуем вычислить по этому правилу $d(3,\ 4)$:

$$z = 0,\quad 4 + 0 = 4\ >\ 3;$$

$$z = 1,\quad 4 + 1 = 5\ >\ 3;$$

$$z = 2,\quad 4 + 2 = 6\ >\ 3$$

.....

Данный процесс будет продолжаться бесконечно, следовательно, $d(3,\ 4)$ не определено.

Таким образом, функция, полученная применением оператора минимизации ко всюду определенной функции, может быть не всюду определенной, а частичной.

Замечание. В случае неопределенности функции в некоторой точке процесс вычисления ее значения с помощью μ-оператора не останавливается, а продолжается неограниченно долго, так же, как и при вычислении ее на машине Тьюринга.

б) Аналогично с помощью оператора минимизации можно получить частичную функцию, выражающую частное от деления двух натуральных чисел.

Дана функция $f(x,\ y) = x \cdot y$. Применим оператор минимизации по переменной x: $x/y = q(x,\ y) = \mu_z\ (y \cdot z = x) =$

$= \mu_z(p(I_2^3(x,\ y,\ z),\ I_3^3(x,\ y,\ z)) = I_1^3(x,\ y,\ z))$.

Задачи

2.1. Используя в качестве исходных функций только константы и простейшие функции, построить примитивно рекурсивные схемы, описывающие нижеследующие функции:

1) sgx; 2) $\overline{sg}x$; 3) nx, где $n \geq 2$ и натуральное;

4) $x \dot{-} 1$; 5) $x \dot{-} y$; 6) x^2; 7) $x^2 + 3$; 8) $x^2 + 2y^2$;

9) $(x+2)^2$; 10) $3x+4y+5z$; 11) 3^{x+2}; 12) $sg\ (2^x)$;

13) $2^x \dot{-} y$; 14) $x^y + 5$; 15) $x! = \begin{cases} 1, & если\ \ x = 0 \\ 1 \cdot 2 \cdot 3 \ldots \cdot x, & если\ \ x \geq 1 \end{cases}$;

16) (x+2)!; 17) $\left[\frac{x}{2}\right]$;

18) $x \oplus y = \begin{cases} 0, & если\ \ x + y - четное \\ 1, & если\ \ x + y - нечетное \end{cases}$ (сумма по модулю 2).

2.2. Применяя операцию примитивной рекурсии к функциям $g(x)$ и $h\ (x,\ y,\ z)$ по переменной y, построить функцию $f(x,y) = R\ (g,h)$, записав ее в «аналитической» форме:

1) $g(x) = 2x$, $h\ (x,\ y,\ z) = x + 2z$;

2) $g(x) = x^2, \quad h\ (x,\ y,\ z) = x^2 + 2y;$

3) $g(x) = 2^x, \quad h\ (x,\ y,\ z) = 2^x \cdot y;$

4) $g(x) = 2^x, \quad h\ (x,\ y,\ z) = 2^x \cdot z\ ;$

5) $g(x) = 2^x, \quad h\ (x,\ y,\ z) = 2^y \cdot z\ ;$

6) $g(x) = 3^x, \quad h\ (x,\ y,\ z) = 3^x \cdot z;$

7) $g(x) = 3^x, \quad h\ (x,\ y,\ z) = 3^y \cdot z;$

8) $g(x) = 3^x, \quad h\ (x,\ y,\ z) = 3^x \cdot y;$

9) $g(x) = 3^x, \quad h\ (x,\ y,\ z) = 3^z;$

10) $g(x) = 1, \quad h\ (x,\ y,\ z) = x \dot{-} y;$

11) $g(x) = 2, \quad h\ (x,\ y,\ z) = z \dot{-} x;$

12) $g(x) = 2x, \quad h\ (x,\ y,\ z) = \begin{cases} y, & \textit{если} \quad x \geq y \\ 0, & \textit{если} \quad x < y \end{cases};$

13) $g(x) = sgx, \quad h\ (x,\ y,\ z) = x \cdot sgy + z \cdot \overline{sg}x.$

2.3. Применить операцию минимизации к функции одной переменной $h(x)$ по переменной x (результирующую функцию представить в «аналитической» форме):

1) $h(x) = 3;$ 2) $h(x) = x + 2;$

3) $h(x) = x \dot{-} 2;$ 4) $h(x) = x - 2;$

5) $h(x) = 2x + 1;$ 6) $h(x) = 2x \dot{-} 1;$

7) $h(x) = 2x - 1;$ 8) $sg\ (x + 3);$

9) $sg\ (x \dot{-} 3);$ 10) $\overline{sg}\ (x + 3);$

11) $\overline{sg}\ (x \dot{-} 3);$ 12) $h(x) = 5x \dot{-} 2;$

13) $h(x) = 5x + 2;$ 14) $h(x) = 5x - 2;$

15) $h(x) = \left[\frac{x}{5}\right];$ 16) $h(x) = \frac{x}{5};$

17) $h(x) = \left[\frac{x}{2}\right] - \left[\frac{x}{3}\right];$ 18) $h(x) = x \dot{-} \left[\frac{x}{2}\right];$ 19) $h(x) = \left[\frac{x \dot{-} 1}{2}\right];$

20) $h(x) = \begin{cases} x + 1, & \text{если} \quad x = 0,\ 1,\ 2, \\ \text{не определено}, & \text{если} \quad x = 3 \\ x - 4, & \text{если} \quad x \geq 4 \end{cases}.$

2.4. Применить операцию минимизации к функции многих переменных $h(x_1,\ x_2)$ по переменной x_i (результирующую функцию представить в «аналитической» форме):

1) $h(x_1,\ x_2) = \mathrm{x}_1 + x_2 + 1, i = 1,\ 2;$

2) $h(x_1,\ x_2) = \mathrm{x}_1 \dot{-} x_2, i = 1,\ 2;$

3) $h(x_1,\ x_2) = I_1^2(x_1,\ x_2), i = 1,\ 2;$

4) $h(x_1,\ x_2) = sg\ (x_1 \dot{-} 2x_2), i = 1,\ 2;$

5) $h(x_1,\ x_2) = x_1 \dot{-} \frac{1}{x_2}, i = 1,\ 2.$

6) $h(x_1,\ x_2) = 2^{x_1}(2x_2 + 1), \qquad i = 1,\ 2;$

2.5. Найти примитивно рекурсивную функцию (если она существует), из которой однократным применением операции минимизации можно получить частично рекурсивную функцию:

1) $f(x_1) = 2 - \mathrm{x}_1;$ 2) $f(x_1) = \frac{\mathrm{x}_1}{2};$ 3) $f(x_1) = \frac{1}{x_1+1};$

4) $f(x_1) = sg\ (\mathrm{x}_1 - 1);$ 5) $f(x_1, x_2) = x_1 - 2x_2;$

6) $f(x_1,\ x_2) = \frac{x_1 - x_2}{3};$ 7) $f(x_1,\ x_2) = \frac{x_1}{x_2+2};$

8) $f(x_1,\ x_2) = \frac{x_1}{1 - x_1 \cdot x_2}.$

2.6. Докажите следующие свойства усеченной разности: а) $0 \dot{-} \mathrm{y} = 0;$

б) $x \dot{-} y = s(x) \dot{-} s(y);$ в) $x + (y \dot{-} x) = y + (x \dot{-} y);$

г) $x \dot{-} (y + z) = (x \dot{-} y) \dot{-} z;$ д) $(x \dot{-} y) \dot{-} z = (x \dot{-} z) \dot{-} y.$

Решение: г) Доказательство проведем индукцией по y. При $\mathrm{y} = 0$, очевидно, имеем: $x \dot{-} (0 + z) = \mathrm{x} \dot{-} z = (x \dot{-} 0) \dot{-} z.$ Предположим, что утверждение верно для y:

$x \dot{-} (y + z) = (x \dot{-} y) \dot{-} z.$

Докажем, что оно верно для $y + 1$:

$x \dot{-} ((y + 1) + z) = (x \dot{-} (y + 1)) \dot{-} z$ $(*)$.

Последнее утверждение, в свою очередь, докажем индукцией по x. Базисное утверждение этой индукции $0 \dot{-} ((y + 1) + z) = (0 \dot{-} (y + 1)) \dot{-} z$

верно в силу свойства а). Предположим теперь, что утверждение верно для x: $x \dot{-} ((y+1)+z) = (x \dot{-} (y+1)) \dot{-} z$.

Докажем, что оно верно для $x+1$: $(x+1) \dot{-} ((y+1)+z) = ((x+1) \dot{-} (y+1)) \dot{-} z$.

Вычисляем: $(x+1) \dot{-} ((y+1)+z) = (x+1) \dot{-} ((y+z)+1) = s(x) \dot{-} s(y+z) =$ (по свойству б)) $= ((x+1) \dot{-} (y+1)) \dot{-} z$.

Индукция по x завершена, чем доказано утверждение $(*)$. Последнее, в свою очередь, завершает индукцию по y, что и доказывает исходное утверждение.

2.7. Доказать, что следующие функции примитивно рекурсивны, для чего представьте их в виде суперпозиции сложения $sum\ (x,\ y) = x + y$ и усеченной разности $x \dot{-} y$:

1) $|x - y|$; 2) $min\ \ (x,\ y)$; 3) $max\ \ (x,\ y)$.

2.8. Доказать, что если функция $f(x_1,\ x_2,\ x_3,\ x_4)$ – примитивно рекурсивна, то следующие функции примитивно рекурсивны:

а) $g(x_1,\ x_2,\ x_3,\ x_4) = f(x_2,\ x_1,\ x_3,\ x_4)$ – (перестановка аргументов);

б) $\phi\ (x_1,\ x_2,\ x_3,\ x_4) = f(x_2,\ x_3,\ x_4,\ x_1)$ – (циклическая перестановка аргументов);

в) $h\ (x_1,\ x_2,\ x_3,\ x_4,\ x_5) = f(x_1,\ x_2,\ x_3,\ x_4)$ – (введение фиктивного аргумента);

г) $\psi\ (x_1,\ x_2,\ x_3) = f(x_1,\ x_1,\ x_2,\ x_3)$ – (отождествление аргументов).

2.9. Доказать, что следующие функции частично рекурсивны:

а) нигде не определенная функция *w*, т. е. функция с пустой областью определения;

б) $f(x,y) = \begin{cases} x - y, & \text{если } x \geq y, \\ \text{не определена} & \text{в остальных случаях;} \end{cases}$

в) $f(x,y) = \begin{cases} \frac{x}{y}, & \text{если } y \text{ делит } x, \\ \text{не определена} & \text{в остальных случаях;} \end{cases}$

г) функция, определенная в конечном числе точек.

Раздел 3. Функции, вычислимые по Тьюрингу

3.1. Машина Тьюринга. Функции, вычислимые по Тьюрингу

Машина Тьюринга – математическое уточнение понятия алгоритма, названное по имени английского математика, сформулировавшего его в 1937 году, за девять лет до появления первой электронно-вычислительной машины.

Машина Тьюринга (МТ) представляет собой абстрактное устройство, состоящее из ленты, считывающей (и печатающей) указателя и управляющего устройства.

Бесконечная в обе стороны *лента* разбита на ячейки (клетки). Во всякой ячейке в каждый дискретный момент времени находится в точности один символ из *внешнего алфавита* $A = \{a_0,\ a_1, \dots,\ a_n\}$ $(n \geq 1)$. Некоторый символ a_0 алфавита A называется пустым, а любая ячейка, содержащая в данный момент пустой символ, называется пустой ячейкой (в этот момент). В качестве пустого символа обычно используют 0 (нуль) или λ.

Словом в алфавите A (или в любом другом алфавите) называется любая последовательность букв соответствующего алфавита.

Лента предполагается потенциально неограниченной в обе стороны. Это следует понимать так: в каждый момент времени лента конечна (т. е. содержит конечное число ячеек), но «размеры» ленты (число ячеек на ней) при необходимости можно увеличивать.

	...	a_{i_1}	a_{i_2}	a_{i_3}	a_{i_4}	a_{i_5}	...	

Управляющее устройство в каждый момент времени находится в некотором *состоянии* q_j, принадлежащем множеству $Q = \{q_0,\ q_1, \dots,\ q_m\}$. Множество Q называется *внутренним алфавитом* (или множеством внутренних состояний).

Замечание. В дальнейшем, если не оговаривается противное, считаем, что $|Q| \geq 2$, q_1 – начальное состояние, а q_0 – заключительное состояние.

Считывающий (и пишущий) указатель перемещается вдоль ленты так, что в каждый момент времени он обозревает ровно одну ячейку ленты. В одном

такте работы МТ указатель может сдвигаться только на одну клетку (вправо R, влево L) или оставаться на месте (E). Указатель считывает содержимое обозреваемой ячейки и записывает в нее вместо обозреваемого символа некоторый символ из внешнего алфавита. «Засылаемый» в ячейку символ может, в частности, совпадать с тем, который обозревался (в данный момент).

В процессе работы управляющее устройство в зависимости от состояния, в котором оно находится, и символа, обозреваемого указателем, изменяет свое внутреннее состояние или остается в прежнем состоянии, выдает указателю команду напечатать в обозреваемой ячейке определенный символ из внешнего алфавита и «приказывает» указателю либо остаться на месте, либо сдвинуться на одну ячейку влево, либо сдвинуться на одну ячейку вправо.

Такое действие устройства управления называют *командой*, которая записывается следующим образом: $q_i a_j \rightarrow q_k a_l S$, где q_i – внутреннее состояние машины МТ в данный момент, a_j – считываемый в этот момент символ, q_k – внутреннее состояние машины МТ в следующий момент (может быть q_k=q_i), a_l – символ, на который изменяется символ a_j (в частности, оставляет его без изменения, т. е. $a_l = a_j$); S– символ сдвига указателя МТ: L (влево), R (вправо), E (на месте). Выражения $q_i a_j$ и $q_k a_l S$ называются левой и правой частями команды. В левой части ни одной команды q_0 не встречается. Всех команд, в которых левые части попарно различны, конечное число (так как множества $Q \backslash \{q_0\}$ и A конечны) и их совокупность называется *программой* МТ.

Определение 3.1. *Программа МТ* – система команд вида $q_i a_j \rightarrow q_k a_l S$, где q_i (соответственно a_j) – символы внутреннего (соответственно внешнего) алфавитов, S– символ сдвига указателя МТ: L (влево), R (вправо), E (на месте); $q_i \ , \ q_k \in Q, (1 \leq i \leq m, 0 \leq k \leq m), \ a_j, \ a_l \in A$.

Программа полностью определяет работу МТ, поэтому можно сказать, что МТ задана, если задана ее программа.

Программу МТ можно задавать в виде таблицы (1), которую называют *тьюринговой функциональной схемой*.

Таблица 1

		a_1	…	a_j	…	a_n
q_1					⋮	
q_2				⋮		
⋮				⋮		
q_i	…	…	…	$q_k a_l S$	…	…
⋮				⋮		
q_m				⋮		

В клетке таблицы на пересечении i -й строки и j -го столбца вписана правая часть команды $q_k a_l S$, соответствующая левой части той же команды $q_i a_j$. Если в программе МТ команда с левой частью $q_i a_j$ отсутствует, то в таблице на пересечении строки q_i и столбца a_j ставится прочерк.

Работа МТ полностью определяется ее программой, т. е. две МТ с общей функциональной схемой неразличимы, и различные машины Тьюринга имеют разные программы.

Определение 3.2. *Машина Тьюринга* (МТ*)* – система, задаваемая тройкой $T = (A,\ Q,\ P)$, которая удовлетворяет следующим условиям:

1) множества $A = \{a_1,\ a_2, \dots,\ a_n\}$ – внешний алфавит МТ и $Q = \{q_0,\ q_1, \dots,\ q_m\}$ – внутренний алфавит состояний управляющего устройства конечны, не пересекаются и не содержат букв L, R, E;

2) пустой символ $a_0 \in A$; начальное и заключительное состояния $q_1,\ q_0 \in Q$;

3) P– такая программа с внешним алфавитом A и внутренним алфавитом Q, что

   a) не существует в P двух различных команд вида $q_i a_j \to q_k a_l S$ с одинаковыми левыми частями;

   b) q_0 не входит в левую часть ни одной команды из *P*.

4) МТ – воображаемое устройство, состоящее из управляющего устройства, реализующего функционирование МТ в дискретном времени по программе *P*; ленты, разделенной на ячейки; читающего-пишущего указателя.

Если левую часть команды $q_i a_j$ записать в виде пары $(q_i,\ a_j)$, а правую $q_k a_l S$ в виде тройки $(q_k,\ a_l,\ S)$, где $a_j,\ a_l \in A,\ q_i \in Q\backslash\{q_0\},\ q_k \in Q,\ S \in \{R,\ L,\ E\} = D$, то, используя понятие произведения множеств, можно утверждать, что МТ представляет собой функцию с областью определения $Q\backslash\{q_0\} \times A$, принимающую значения из множества $Q \times A \times D$, или отображение первого множества на второе: $Q\backslash\{q_0\} \times A \to Q \times A \times D$ [так как каждой паре $(q_i,\ a_j)$ соответствует единственная тройка $(q_k,\ a_l,\ S)$].

МТ функционирует в дискретном времени $t = 0,1,2,\ldots$ Выполнение одной команды называется шагом. Вычисление (или работа) МТ есть последовательность шагов одного за другим без пропусков, начиная с первого.

Работа МТ начинается с задания в первый (начальный) момент:

1) слова на ленте, т. е. последовательности символов из алфавита A, записанных в ячейках ленты слева направо;
2) положения считывающего указателя;
3) внутреннего состояния машины. Совокупность этих трех условий (в данный момент времени) называется конфигурацией (в данный момент времени).

Определение 3.3. *Конфигурацией МТ* (машинным словом) называется слово К в алфавите $A \cup Q$ вида Uq_iV, где U – слово на ленте левее указателя, V– слово правее слова U, начиная с символа, находящегося в той ячейке, на которую указывает указатель.

		u_{i_1}	u_{i_2}	v_{i_3}	v_{i_4}	v_{i_5}		
				$\uparrow q_j$				

Конфигурацию в данный момент времени можно задать следующим словом: $K = u_{i_1} u_{i_2} q_j v_{i_3} v_{i_4} v_{i_5}$, здесь U=$u_{i_1} u_{i_2}$ и $V = v_{i_3} v_{i_4} v_{i_5}$.

Стандартная начальная конфигурация – q_1V, а *стандартная заключительная конфигурация* – q_0V, это значит, что в начале (в конце) работы считывающий указатель находится в начальном (заключительном) состоянии в самой левой из заполненных ячеек.

Если при выполнении некоторой команды машины T из конфигурации K_i получается конфигурация K_{i+1}, то K_{i+1} называется *непосредственно выводимой* из K_i (обозначение: T: $K_i \to K_{i+1}$ или $K_i|{=}K_{i+1}$).

Если K_1 – начальная конфигурация, то последовательность $K_1 \to K_2 \to \cdots \to K_m$, где $K_i|{=}K_{i+1}$ при $1 \le i \le m-1$, называется *тьюринговым вычислением*. При этом говорят, что конфигурация K_m *выводима* из конфигурации K_1, и пишут $K_1|{-}K_m$. Если K_m является к тому же заключительной конфигурацией, то говорят, что K_m *заключительно выводима* из K_1, и пишут $K_1|{\div}K_m$. Если T: $K_1 \to K_2 \to \cdots \to K_m \to \cdots$ (т. е. МТ работает бесконечно), то пишут T: $K_1 \to$.

Определение 3.4. *Протокол работы МТ* – детерминированная (конечная или бесконечная) последовательность конфигураций K_{i_1}, K_{i_2}, ..., такая, что T: $K_{i_r} \to K_{i_{r+1}}$, т. е. при выполнении некоторой команды машины T из конфигурации K_{i_r} получается конфигурация $K_{i_{r+1}}$.

Замечание: как в начальный, так и в каждый из последующих шагов работы МТ конфигурация на ленте остается конечной, так как это слово в алфавите $A \cup Q$.

Работа МТ считается законченной в двух случаях:

1) в некоторый момент времени выполняется команда, правая часть которой содержит q_0;

2) в программе нет ни одной команды, соответствующей конфигурации, полученной на некотором шаге.

В остальных случаях будем считать работу МТ бесконечной.

Определение 3.5. Говорят, что *машина T применима к слову R*, если, начав работу на слове R, МТ T остановится через конечное число шагов; если при этом получается слово Q, то пишут $\mathrm{T}(R) = Q$. Если T не останавливается, то машина T *не применима* к слову Q.

МТ считается *применимой* к множеству слов $\{R_i\}$, если она применима к каждому слову R_i.

Зоной работы машины T (на слове R) называется множество всех ячеек, которые за время работы машины хотя бы один раз обозреваются указателем.

Часто будет использоваться обозначение $[P]^m$ для слов вида $PP\ldots P$ (m раз), где $m \geq 0$; при $m = 0$ считаем, что $[P]^m$ – пустое слово; если $P = a$ – слово длины 1, то вместо $aa\ldots a$ (m раз) и $[a]^m$ будем писать a^m.

Через W будем обозначать произвольное конечное слово во внешнем алфавите машины Тьюринга (в частности, пустое, т. е. состоящее из пустых символов внешнего алфавита).

При описании работы машины Тьюринга «на языке конфигураций» будут использоваться выражения, аналогичные такому: $q_1 1^x 0 1^y 0 W | \dot{-} 1^y 0 q_0 W$, $x \geq 1$ и $y \geq 1$.

Приведенное выражение надо понимать так: машина «стирает» слово 1^x, проходит без изменения слово 1^y и останавливается на первой букве слова W; если же W – пустое слово, то «останов» происходит на втором 0 (нуле) после слова 1^y.

Пример 3.1. Дана МТ с внешним алфавитом $A = \{0,1\}$ (здесь 0 – символ пустой ячейки), алфавитом внутренних состояний $Q = \{q_0, q_1, q_2\}$ и со следующей функциональной схемой (программой): $q_1 0 \to q_2 0 L$

$$q_2 0 \to q_0 1 E$$

$$q_1 1 \to q_1 1 L$$

$$q_2 1 \to q_2 1 L$$

Выпишем последовательно конфигурации машины при переработке слова $R = 101$, исходя из стандартного начального положения

(1)

		1	0	1		

$\uparrow q_1$

Замечание: слева и справа от слова на ленте располагаются пустые символы, которые обозначаются в том случае, если в ячейке с пустым символом находится указатель МТ.

На первом шаге действует команда $q_1 1 \to q_1 1L$. В результате получается конфигурация:

(2)

	0	1	0	1		

$\uparrow q_1$

На втором шаге действует команда $q_1 0 \to q_2 0L$. В результате получается конфигурация:

(3)

	0	0	1	0	1		

$\uparrow q_2$

На третьем шаге действует команда $q_2 0 \to q_0 1E$. В результате получается конфигурация:

(4)

	1	0	1	0	1		

$\uparrow q_0$

Эта конфигурация является заключительной, потому что машина оказалась в заключительном состоянии остановки q_0. Таким образом, исходное слово $R = 101$ переработано в слово $Q = 10101$.

Полученную последовательность конфигураций можно записать более коротким способом:

$$K_1 = q_1 101 \to K_2 = q_1 0101 \to K_3 = q_2 00101 \to K_4 = q_0 10101$$

Замечание: слева и справа от слова на ленте располагаются пустые символы, которые обозначаются в том случае, если в ячейке с пустым символом находится указатель МТ.

Пример 3.2. Выяснить, применима ли МТ, задаваемая программой P, к слову W (исходя из стандартного положения). Если применима, то выписать результат применения МТ к слову W.

$$P: \begin{cases} q_1 0 \to q_2 1R \\ q_1 1 \to q_1 0L \\ q_2 0 \to q_3 1R \\ q_2 1 \to q_3 0L \\ q_3 0 \to q_1 0R \end{cases}$$

а) $W = 10^3 1$; б) $W = [10]^2 1$.

Решение: а) Исходя из конфигурации $K_1 = q_1 10^3 1$

(1)

		1	0	0	0	1	

$\uparrow q_1$

получаем последовательно следующие конфигурации:

(2) $K_2 = q_1 0^5 1$

	0	0	0	0	0	1	

$\uparrow q_1$

(3) $K_3 = 1 q_2 0^4 1$

	1	0	0	0	0	1	

$\uparrow q_2$

(4) $K_4 = 1^2 q_3 0^3 1$

	1	1	0	0	0	1	

$\uparrow q_3$

(5) $K_5 = 1^2 0 q_1 0^2 1$

	1	1	0	0	0	1	

$\uparrow q_1$

(6) $K_6 = 1^2 01 q_2 01$

	1	1	0	1	0	1	

$\uparrow q_2$

(7) $K_7 = 1^2 01^2 q_3 1$

	1	1	0	1	1	1	

$\uparrow q_3$

Так как команды вида $q_3 1 \to q_i a S$ в программе P нет, то последняя конфигурация (т. е. $K_7 = 1^2 01^2 q_3 1$) заключительная. Следовательно, машина T к слову $W = 10^3 1$ применима и $T(W) = 1^2 01^3$.

б) Исходя из конфигурации $K_1 = q_1 [10]^2 1$ или $\underset{q_1}{1}\ 0\ 1\ 0\ 1$, получаем последовательно следующие конфигурации:

Замечание: для простоты изображения различных конфигураций МТ будем в дальнейшем записывать информацию в виде слова, не изображая ленты и ее разбивки на ячейки, а вместо изображения управляющего указателя и состояния машины записывать только состояние.

$K_2 = q_1 0^3 101$ или $\underset{q_1}{0}\ 0\ 0\ 1\ 0\ 1$

$K_3 = 1 q_2 0^2 101$ или $1\ \underset{q_2}{0}\ 0\ 1\ 0\ 1$

$K_4 = 1^2 q_3[01]^2$ или $1\ 1\ \underset{q_3}{0}\ 1\ 0\ 1$

$K_5 = 1^2 0 q_1 101$ или $1\ 1\ 0\ \underset{q_1}{1}\ 0\ 1$

$K_6 = 1^2 q_1 0^3 1$ или $1\ 1\ \underset{q_1}{0}\ 0\ 0\ 1$

$K_7 = 1^3 q_2 0^2 1$ или $1\ 1\ 1\ \underset{q_2}{0}\ 0\ 1$

$K_8 = 1^4 q_3 01$ или $1\ 1\ 1\ 1\ \underset{q_3}{0}\ 1$

$K_9 = 1^4 0 q_1 1$ или $1\ 1\ 1\ 1\ 0\ \underset{q_1}{1}$

$K_{10} = 1^4 q_1 0$ или $1\ 1\ 1\ 1\ \underset{q_1}{0}\ 0$

$K_{11} = 1^5 q_2 0$ или $1\ 1\ 1\ 1\ 1\ \underset{q_2}{0}$

$K_{12} = 1^6 q_3 0$ или $1\ 1\ 1\ 1\ 1\ 1\ \underset{q_3}{0}$

$K_{13} = 1^6 0 q_1 0$ или $1\ 1\ 1\ 1\ 1\ 1\ 0\ \underset{q_1}{0}$

$K_{14} = 1^6 01 q_2 0$ или $1\ 1\ 1\ 1\ 1\ 1\ 0\ 1\ \underset{q_2}{0}$

$K_{15} = 1^6 01^2 q_3 0$ или $1\ 1\ 1\ 1\ 1\ 1\ 0\ 1\ 1\ \underset{q_3}{0}$

$K_{16} = 1^6 01^2 0 q_1 0$ или $1\ 1\ 1\ 1\ 1\ 1\ 0\ 1\ 1\ 0\ \underset{q_1}{0}$

$K_{17} = 1^6 01^2 01 q_2 0$ или $1\ 1\ 1\ 1\ 1\ 1\ 0\ 1\ 1\ 0\ 1\ \underset{q_2}{0}$

$K_{18} = 1^6 01^2 01^2 q_3 0$ или $1\ 1\ 1\ 1\ 1\ 1\ 0\ 1\ 1\ 0\ 1\ 1\ \underset{q_3}{0}$

и т. д....

Ясно, что этот процесс продолжается неограниченно. Значит, машина T к слову $R = [10]^2 1$ не применима.

Пример 3.3. Построим МТ, которая применима ко всем словам в алфавите $\{a,\ b\}$ и делает следующее: любое слово $x_1 x_2 \dots x_n$, где $x_i \in \{a,\ b\}$ для всех $1 \le i \le n$, преобразует в слово $x_2 \dots x_n x_1$.

Решение. За внешний алфавит МТ возьмем $A = \{\lambda, a, b\}$, внутренний алфавит $Q = \{q_0,\ q_1,\ q_2,\ q_3,\ q_4\}$, программу зададим таблицей:

	q_1	q_2	q_3	q_4
	—	q_4aE	q_4bE	$q_0\lambda R$
a	$q_2\lambda R$	q_2aR	q_3aR	q_4aL
b	$q_3\lambda R$	q_2bR	q_3bR	q_4bL

Работу МТ можно описать следующим образом: она запоминает первую букву исходного слова (при этом стирая ее), и указатель движется вправо до первой пустой клетки, в которую и записывается первая буква.

Рассмотрим работу МТ над словами: а) $R = ba$; б) $R = abb$.

а) Начальная конфигурация $\underset{q_1}{b}\ a$

$a\ \underset{q_3}{\lambda}$

$a\ b\ \underset{q_4}{\lambda}$

$a\ \underset{q_4}{b}\ a$

$\underset{q_4}{\lambda}\ a\ b\ a$

$a\ \underset{q_0}{b}\ a$

б) $\underset{q_1}{a}\ b\ b$

$b\ \underset{q_2}{b}$

$b\ b\ \underset{q_2}{\lambda}$

$b\ b\ a\ \underset{q_4}{\lambda}$

$b\ \underset{q_4}{b}\ a$

$\underset{q_4}{b}\ b\ a$

$\underset{q_4}{\lambda}\ b\ b\ a$

$b\ b\ a\ \underset{q_0}{\lambda}$

а) $K_1 = q_1ba \to q_3a \to aq_3\lambda \to aq_4b \to q_4ab \to$

$\to q_4\lambda ab \to q_0ab$ – заключительная конфигурация.

б) $q_1abb \to q_2bb \to bq_2b \to bbq_2\lambda \to bq_4ba \to$

$\to q_4bba \to q_4\lambda bba \to q_0bba.$

Определим вычисление функций на машинах Тьюринга.

Будем рассматривать словарные частичные функции $f: A^* \to A^*$, где A^*– множество слов конечной длины в алфавите A.

Определение 3.6. Говорят, что машина Тьюринга Т *правильно вычисляет частичную словарную функцию* f, если для любого слова $P \in A^*$ выполнено:

1) если $f(P)$ определено и $f(P) = Q$, то МТ применима к начальной конфигурации q_1P и заключительной конфигурацией является q_0Q;

2) если $f(P)$ не определено, то машина T не применима к начальной конфигурации q_1P.

Аналогично можно определить вычисление числовых функций на машинах Тьюринга. Под числовыми функциями будем понимать функции, значениями которых и значениями их аргументов являются неотрицательные целые числа. В дальнейшем рассматриваются, как всюду определенные функции, у которых $D_f = N_0$, так и частичные числовые функции, у которых $D_f \subset N_0$. Например, функция $\frac{y}{2}$ определена только для четных целых положительных чисел, а для остальных не определена; $3 - \frac{y}{2}$ определена только для $y = 0,2,4,6$, а для остальных значений y не определена; функция $\frac{y^2}{3-\frac{y}{2}}$ определена для $y = 0,2,4$.

При вычислении числовых функций на машинах Тьюринга пользуются специальным кодированием чисел: натуральному числу n ставят в соответствие набор из $n + 1$ единиц, который обозначают через 1^{n+1}. Слово 1^{n+1} называется *основным машинным кодом* или просто *кодом* числа n

Таблица кодирования

число	код числа	сокращенная запись
0	1	1
1	11	1^2
2	111	1^3
...	...	...
n	11...1	1^{n+1}
...	...	...

Набор чисел$x_1, x_2, \dots, x_n$ будем записывать в виде слова $1^{x_1+1} * 1^{x_2+1} * \dots * 1^{x_n+1}$ (код набора чисел в алфавите $\{\lambda,*,1\}$) или $1^{x_1+1}01^{x_2+1}0 \dots 01^{x_n+1}$ (код набора чисел в алфавите $\{0,1\}$).

Определение 3.7. Числовая функция $f(x_1,\ x_2, \dots,\ x_n)$ называется *вычислимой по Тьюрингу*, если существует машина Тьюринга такая, что для любых $m,\ m_1,\ m_2, \dots,\ m_n \in N_0$, если при $x_1 = m_1,\ x_2 = m_2,\ \dots,\ x_n = m_n$ имеем $f(m_1,\ m_2, \dots,\ m_n) = m$, то эта машина применима к слову $1^{m_1+1} * 1^{m_2+1} * \dots * 1^{m_n+1}$ и в заключительной конфигурации на некотором участке ленты будет слово 1^{m+1}, а остальные клетки окажутся пустыми. Если же $f(m_1,\ m_2, \dots,\ m_n)$ не определено, то эта машина либо не применима к слову $1^{m_1+1} * 1^{m_2+1} * \dots * 1^{m_n+1}$, либо в заключительной конфигурации получается слово, которое не является кодом никакого числа из N_0.

Если функция f вычислима по Тьюрингу с помощью машины T_f, то будем говорить, что машина T_f *вычисляет функцию* f.

Определение 3.8. Говорят, что машина Тьюринга T *правильно вычисляет функцию* $f(x_1,\ x_2, \dots,\ x_n)$, если:

1) в случае, когда $f(m_1, m_2, \dots, m_n) = m$ определено, МТ, начав работу с левой единицы кода $1^{m_1+1} * 1^{m_2+1} * \dots * 1^{m_n+1}$, останавливается, и указатель машины в заключительной конфигурации обозревает левую единицу кода 1^{m+1}.
2) в случае, когда $f(m_1, m_2, \dots, m_n)$ не определено, МТ, начав работу с левой единицы кода $1^{m_1+1} * 1^{m_2+1} * \dots * 1^{m_n+1}$, не останавливается.

Функция называется *правильно вычислимой*, если есть МТ, которая ее правильно вычисляет.

Следующие примеры показывают, как строятся машины Тьюринга, реализующие некоторые простые арифметические алгоритмы.

Пример 3.4. Построим МТ с внешним алфавитом $A = \{\lambda, 1\}$, правильно вычисляющую функцию $o(x) = 0$.

Этот алгоритм может быть сформулирован в виде следующего словесного предписания: «Сотри у данного слова (последовательности единиц) все единицы, кроме одной (соответствует нулю), и остановись. Полученное слово и есть результат».

Действие стирания единицы можно задать с помощью команды $q_1 1 \to q_1 \lambda R$. Алгоритм, заданный одной этой командой, будет стирать по одной единице у исходного слова, пока управляющий указатель не будет указывать на пустую ячейку, тогда действие этой команды закончится. Останется применить заключительную команду $q_1 \lambda \to q_0 1E$, которая в пустую ячейку ставит единицу, соответствующую числу 0, и завершает работу (т. к. в правой части команды стоит заключительное состояние q_0). Таким образом, программа МТ, вычисляющая нулевую функцию, состоит из двух команд: $P: \begin{cases} q_1 1 \to q_1 \lambda R \\ q_1 \lambda \to q_0 1E \end{cases}$, применима к каждому слову в алфавите A, и работу МТ при вычислении $o(x) = 0$ можно представить в виде следующей последовательности конфигураций:

$$q_1 1^{x+1} \to \ q_1 1^x \to \ q_1 1^{x-1} \to \cdots \to q_1 1^2 \to q_1 1 \to q_1 \lambda \to q_0 1.$$

Пример 3.5. Построим МТ с внешним алфавитом $A = \{\lambda, 1\}$, правильно вычисляющую функцию $s(x) = x + 1$.

Решение: Рассматриваемый алгоритм – это алгоритм перехода от произвольного числа к числу на единицу больше, или алгоритм, перерабатывающий слово, изображающее число x в слово, изображающее $x + 1$. Этот алгоритм может быть сформулирован в виде следующего словесного предписания: «Припиши к данному слову (последовательности единиц) еще одну единицу и остановись. Полученное слово и есть результат».

Рассмотрим два способа реализации указанного алгоритма в зависимости от того, где будем приписывать дополнительную единицу: а) в начале слова; б) в конце слова.

а) $P_1: \begin{cases} q_1 1 \to q_1 1L \text{ (сдвиг влево, оставляя 1 без изменения)} \\ q_1 \lambda \to q_0 1E \quad \text{(запись 1 в пустую ячейку и останов)} \end{cases}$.

Соответствующая последовательность конфигураций: $q_1 1^{x+1} \to q_1 \lambda 1^{x+1} \to q_0 1^{x+2}$.

Данная МТ P_1 применима ко всем $x \in N_0$.

В частном случае работа P_1 при вычислении $s(2)$ состоит из конфигураций:

$K_1 = q_1 111$ или $\underset{q_1}{1}\ 1\ 1$

$K_2 = q_1 \lambda 111$ или $\underset{q_1}{\lambda}\ 1\ 1\ 1$

$K_3 = q_0 1111$ или $\underset{q_0}{1}\ 1\ 1\ 1$

б) P_2: $\begin{cases} q_1 1 \to q_1 1R \ \ (\textit{цикл движения вправо до конца слова}) \\ q_1 \lambda \to q_2 1L \ (\textit{запись } 1 \textit{ в пустую ячейку и сдвиг влево}) \\ q_2 1 \to q_2 1L \ \ \ (\textit{цикл движения влево к началу слова}) \\ q_2 \lambda \to q_0 \lambda R \qquad\qquad (\textit{конец цикла, останов}) \end{cases}$

Соответствующая последовательность конфигураций:

$$q_1 1^{x+1} \to 1 q_1 1^x \to \cdots \to 1^x q_1 1 \to 1^{x+1} q_1 \lambda \to 1^{x+1} q_2 1 \to$$

$$\to 1^x q_2 1^2 \to \cdots \to q_2 1^{x+2} \to q_2 \lambda 1^{x+2} \to q_0 1^{x+2}.$$

В частном случае работа МТ P_2 при вычислении $s(2)$ состоит из конфигураций:

$K_1 = q_1 111$ или $\underset{q_1}{1}\ 1\ 1$

$K_2 = 1 q_1 11$ или $1\ \underset{q_1}{1}\ 1$

$K_3 = 11 q_1 1$ или $1\ 1\ \underset{q_1}{1}$

$K_4 = 111 q_1 \lambda$ или $1\ 1\ 1\ \underset{q_1}{\lambda}$

$K_5 = 11 q_2 11$ или $1\ 1\ \underset{q_2}{1}\ 1$

$K_6 = 1 q_2 111$ или $1\ \underset{q_2}{1}\ 1\ 1$

$K_7 = q_2 1111$ или $\underset{q_2}{1}\ 1\ 1\ 1$

$K_8 = q_2 \lambda 1111$ или $\underset{q_2}{\lambda} 1\ 1\ 1\ 1$

$K_9 = q_0 1111$ или $\underset{q_0}{1}\ 1\ 1\ 1$ – результат

Определение 3.9. Машины Тьюринга T_1 и T_2 называются *эквивалентными* (в алфавите *A*), если для всякого входного слова R (в алфавите *A*) выполняется соотношение $T_1(R) \simeq T_2(R)$, означающее следующее: результаты $T_1(R)$ и $T_2(R)$ определены или не определены одновременно (т. е. машины T_1 и T_2 либо обе применимы, либо обе не применимы к слову R) и, если эти результаты определены, то $T_1(R) = T_2(R)$. Символ $\simeq$ называется знаком условного равенства.

Замечание: в примере 3.5 машины P_1 и P_2 – эквивалентны.

Пример 3.6. Построим МТ с внешним алфавитом

$A = \{\lambda,\ 0,\ 1,\ 2,\ 3, \dots,\ 9\}$, правильно вычисляющую функцию

$s(x) = x + 1$ в десятичной системе счисления.

Решение: Число x будем записывать в десятичной системе на ленте, причем цифры будут помещаться по одной в каждой клетке подряд без пропусков.

Алгоритм решения поставленной задачи может быть записан в виде следующей системы последовательных указаний:

1. Из начального стандартного (показывает на первую цифру числа) положения указателя передвинуть его к последней цифре заданного числа.
2. Стереть последнюю цифру числа и заменить ее цифрой, на единицу большей.
3. Если последняя цифра была меньше цифры 9, то вернуть указатель на начало слова и остановиться.
4. Если последняя цифра была 9, то нужно, стереть цифру 9, записать в освободившуюся клетку цифру 0 и произвести сдвиг влево к соседнему, более высокому разряду, оставаясь в том же внутреннем состоянии. Перейти к следующему шагу 5.
5. Если ячейка, в которую попали в результате сдвига, не пустая, то переходим к указанию 2, иначе к указанию 6.

6. Если в случае сдвига влево управляющий указатель машины вышел на пустую клетку (когда цифры более высокого разряда нет), то машина вписывает в пустую клетку цифру 1. Перейти к указанию 7.
7. Процесс вычисления завершился. Полученное слово является результатом.

Этот алгоритм реализуется в следующей программе:

	q_1	q_2	q_3
λ	$q_2\lambda L$	$q_0 1E$	$q_0\lambda R$
0	$q_1 0R$	$q_3 1L$	$q_3 0L$
1	$q_1 1R$	$q_3 2L$	$q_3 1L$
2	$q_1 2R$	$q_3 3L$	$q_3 2L$
3	$q_1 3R$	$q_3 4L$	$q_3 3L$
4	$q_1 4R$	$q_3 5L$	$q_3 4L$
5	$q_1 5R$	$q_3 6L$	$q_3 5L$
6	$q_1 6R$	$q_3 7L$	$q_3 6L$
7	$q_1 7R$	$q_3 8L$	$q_3 7L$
8	$q_1 8R$	$q_3 9L$	$q_3 8L$
9	$q_1 9R$	$q_2 0L$	$q_3 9L$

Выпишем последовательность конфигураций для чисел:

а) $x = 183$; б) $x = 399$; в) $x = 99$.

а)	б)	в)
$\underset{q_1}{1}\ 8\ 3$	$\underset{q_1}{3}\ 9\ 9$	$\underset{q_1}{9}\ 9$
$1\ \underset{q_1}{8}\ 3$	$3\ \underset{q_1}{9}\ 9$	$9\ \underset{q_1}{9}$
$1\ 8\ \underset{q_1}{3}$	$3\ 9\ \underset{q_1}{9}$	$9\ 9\ \underset{q_1}{\lambda}$
$1\ 8\ 3\ \underset{q_1}{\lambda}$	$3\ 9\ 9\ \underset{q_1}{\lambda}$	$9\ \underset{q_2}{9}$
$1\ 8\ \underset{q_2}{3}$	$3\ 9\ \underset{q_2}{9}$	$\underset{q_2}{9}\ 0$
$1\ \underset{q_3}{8}\ 4$	$3\ \underset{q_2}{9}\ 0$	$\underset{q_2}{\lambda}\ 0\ 0$
$\underset{q_3}{1}\ 8\ 4$	$\underset{q_2}{3}\ 0\ 0$	$\underset{q_0}{1}\ 0\ 0$

$\underset{q_3}{\lambda}\ 1\ 8\ 4 \qquad \underset{q_3}{\lambda}\ 4\ 0\ 0$

$\underset{q_0}{1}\ 8\ 4 \qquad \underset{q_0}{4}\ 0\ 0$

Пример 3.7. Построим МТ с внешним алфавитом $A = \{0,\ 1,\ *\}$, правильно вычисляющую функцию $Sum(x, y) = x + y$.

Решение: Алгоритм сложения целых неотрицательных чисел, перерабатывающий слово, обозначающее систему двух чисел, в слово, обозначающее сумму этих чисел, можно задать предписанием: «Замени звездочку единицей, сотри две лишние единицы по краям слова и остановись. Полученное слово и есть результат», т. е. из начальной конфигурации $K_1 = q_1 1^{x+1} * 1^{y+1}$ нужно получить заключительную конфигурацию $K = q_0 1^{x+y+1}$, где слово $S = 1^{x+y+1}$ – есть результат.

Следующая программа МТ реализует описанный алгоритм:

$$\begin{cases} q_1 1 \to q_2 0R & (\textit{стерли первую единицу заданного слова}) \\ q_2 1 \to q_2 1R & (\textit{цикл движения вправо до } * \text{ и до конца слова}) \\ q_2 * \to q_2 1R & (\textit{заменили } * \text{ на } 1) \\ q_2 0 \to q_3 0L & (\textit{от конца слова начали движение влево}) \\ q_3 1 \to q_4 0L & (\textit{стерли единицу в конце слова}) \\ q_4 1 \to q_4 1L & (\textit{цикл движения влево до начала слова}) \\ q_4 0 \to q_0 0R & (\textit{дошли до начала слова и остановились}) \end{cases}$$

Соответствующая последовательность конфигураций:

$$q_1 1^{x+1} * 1^{y+1} \to q_2 1^x * 1^{y+1} \to \cdots \to 1^x q_2 * 1^{y+1} \to$$

$$\to 1^{x+1} q_2 1^{y+1} \to 1^{x+2} q_2 1^y \to \cdots \to 1^{x+y+2} q_2 0 \to 1^{x+y+1} q_3 1 \to$$

$$\to 1^{x+y} q_4 1 \to 1^{x+y-1} q_4 1^2 \to \cdots \to q_4 0 1^{x+y+1} \to q_0 1^{x+y+1}.$$

Пример 3.8. Построим МТ с внешним алфавитом $A = \{\lambda, 1\}$, правильно вычисляющую функцию

$$sg(x) = \begin{cases} 0, & \textit{если } x = 0 \\ 1, & \textit{если } x \geq 1 \end{cases}.$$

Решение: из начального стандартного положения (указатель в состоянии q_1 показывает на первую цифру числа) в первом такте работы машины можно сдвинуть управляющую головку вправо, стереть первую единицу и перейти в новое состояние q_2. Если в результате сдвига мы попали на пустую ячейку, то,

значит, заданное число было $x = 0$, и нужно, поставив на место пустой клетки 1, перейти в стоп-состояние.

Если же в состоянии q_2 управляющая указатель обозревает символ 1, то $x \neq 0$, поэтому нужно стереть все заданное слово, поставить в конце две единицы, вернуться на одну ячейку назад и остановиться.

Таким образом, программа машины Тьюринга, вычисляющая функцию $sg(x)$, имеет вид:

$$P: \begin{cases} q_1 1 \to q_2 \lambda R \\ q_2 \lambda \to q_0 1 E \\ q_2 1 \to q_3 \lambda R \\ q_3 1 \to q_3 \lambda R \\ q_3 \lambda \to q_4 1 R \\ q_4 \lambda \to q_0 1 L \end{cases}$$

Пример 3.9. *Алгоритм Евклида.*

Алгоритм Евклида сводится к построению убывающей последовательности чисел, из которых первое является большим из двух данных чисел, третье получается, как остаток от деления первого на второе, четвертое – как остаток от деления второго на третье и т. д., пока не будет совершено деление без остатка. Делитель в этом последнем делении и будет результатом решения задачи.

Требуется задать алгоритм Евклида в виде программы машины Тьюринга. Эта программа должна обеспечить попеременное чередование циклов сравнения и циклов вычитания чисел.

Будем использовать внешний алфавит, состоящий из пяти символов $A = \{\lambda, 1, *, \alpha, \beta\}$, здесь α и β – вспомогательные символы – заменители 1, необходимые для написания программы.

Функциональная схема МТ имеет следующий вид:

	λ	$*$	1	α	β	Комментарий
q_1	–	$q_2 * L$	$q_1 1 R$	–	–	начало работы
q_2	$q_4 \lambda R$	$q_2 * L$	$q_3 \alpha R$	$q_2 \alpha L$	$q_2 \beta L$	ставит β
q_3	$q_7 \lambda L$	$q_3 * R$	$q_2 \beta L$	$q_3 \alpha R$	$q_3 \beta R$	ставит β
q_4	$q_{10} \lambda L$	$q_4 \lambda R$	$q_5 * R$	$q_4 \lambda R$	$q_4 1 R$	стираетα,

						заменяетβ
q_5	$q_6 1R$	–	$q_5 1R$	–	–	*ставят справа* 11
q_6	$q_6 1L$	$q_2 * L$	$q_6 1L$	–	–	
q_7	$q_0 \lambda R$	$q_7 \lambda L$	$q_8 * L$	$q_7 1L$	$q_7 \lambda L$	стираетα заменяетβ
q_8	$q_9 1L$	–	$q_8 1L$	–	–	*ставят слева* 11
q_9	$q_9 1R$	$q_2 * L$	$q_9 1R$	–	–	
q_{10}	$q_0 \lambda R$	–	$q_{10} 1L$	–	–	возврат к началу

Пусть для определенности требуется найти *НОД* $(4,\ 6)$.

Начальная конфигурация $K_1 = q_1 1^5 * 1^7$ или $\underset{q_1}{1}1111{*}1111111$.

Дойдем до разделительного символа $*$ и начнем сравнивать числа, изображенные на ленте. С этой целью МТ должна заменять 1, соответствующие первому числу, буквами α, а 1, соответствующие второму числу, буквами β.

$$1\,1\,1\,1\,\underset{q_1}{1} * 1\,1\,1\,1\,1\,1\,1 \rightarrow 1\,1\,1\,1\,\underset{q_2}{1} * 1\,1\,1\,1\,1\,1\,1 \rightarrow$$

$$\rightarrow 1\,1\,1\,1\,\alpha\,\underset{q_3}{*}\,1\,1\,1\,1\,1\,1\,1 \rightarrow 1\,1\,1\,1\,\alpha * \underset{q_3}{1}1\,1\,1\,1\,1\,1 \rightarrow$$

$$\rightarrow 1\,1\,1\,1\,\alpha\,\underset{q_2}{*}\,\beta\,1\,1\,1\,1\,1\,1 \rightarrow 1\,1\,1\,1\,\underset{q_2}{\alpha} * \beta 1\,1\,1\,1\,1\,1 \rightarrow$$

$$\rightarrow 1\,1\,\underset{q_2}{1}1\,\alpha * \beta\,1\,1\,1\,1\,1\,1 \rightarrow 1\,1\,1\,\underset{q_3}{\alpha}\,\alpha * \beta 1\,1\,1\,1\,1\,1 \rightarrow$$

$$\rightarrow 1\,1\,1\,\alpha\,\underset{q_3}{\alpha}\,*\,\beta\,1\,1\,1\,1\,1\,1 \rightarrow 1\,1\,1\,\alpha\alpha\,*\,\underset{q_3}{\beta}1\,1\,1\,1\,1\,1 \rightarrow$$

$$\rightarrow 1\,1\,1\alpha\alpha * \underset{q_3}{\beta}1\,1\,1\,1\,1\,1 \rightarrow 1\,1\,1\alpha\alpha\,*\,\underset{q_2}{\beta}\beta\,1\,1\,1\,1\,1\ldots$$

и т. д. движемся влево в поисках еще не отмеченной ближайшей единицы первого числа, которая должна быть заменена на α. Затем начинается движение управляющего указателя вправо (при этом МТ находится в состоянии q_3) в поисках ближайшей единицы второго числа, которая должна быть заменена на β. На очередном шаге возникнет конфигурация

$q_2 \alpha^5 * \beta^5 11$ или $\underset{q_2}{\alpha}\,\alpha\,\alpha\,\alpha\,\alpha * \beta\,\beta\,\beta\,\beta\,\beta\,1\,1$.

На этом заканчивается цикл сравнения исходных чисел и начинается цикл вычитания, в результате которого меньшее число должно быть стерто целиком;

символы β заменяются на 1 и, следовательно, большее число 6 будет разбито на два числа 4 и 2.

$$\underset{q_4}{\alpha}\alpha\alpha\alpha\alpha * \beta\beta\beta\beta\beta 11 \to^{\lambda} \underset{q_4}{\alpha}\alpha\alpha\alpha * \beta\beta\beta\beta\beta 11 \to \underset{q_4}{\alpha}\alpha\alpha * \beta\beta\beta\beta\beta\ 1\ 1 \to \underset{q_4}{\alpha}\ \alpha\ \alpha\ * \beta\beta\beta\beta\beta\ 1\ 1 \to \underset{q_4}{\alpha}\ \alpha * \beta\ \beta\ \beta\ \beta\ \beta 1\ 1 \to \underset{q_4}{\alpha} * \beta\ \beta\ \beta\beta\ 1\ 1 \to$$

$$\to \underset{q_4}{*}\ \beta\ \beta\ \beta\ \beta\ \beta\ 1\ 1 \to \underset{q_4}{\beta}\ \beta\ \beta\ \beta\ \beta\ 1\ 1 \to$$

$$\to 1\ \underset{q_4}{\beta}\ \beta\ \beta\ \beta\ 1\ 1 \to 1\ 1\ \underset{q_4}{\beta}\ \beta\ \beta\ 1\ 1 \to 1\ 1\ 1\ \underset{q_4}{\beta}\beta 11 \to$$

$$\to 1111\underset{q_4}{\beta}11 \to 11111\underset{q_4}{1}1 \to 11111 * \underset{q_5}{1} \to 111\ 1\ 1 * 1\underset{q_5}{\lambda} \to$$

$$\to 11\ 11\ 1 * 11\underset{q_6}{\lambda} \to 1\ 1\ 1\ 1\ 1 * 1\underset{q_6}{1}1 \to 1\ 1\ 1\ 1\ 1 * \underset{q_6}{1}1\ 1 \to$$

$$\to 1\ 1\ 1\ 1\ 1\ \underset{q_6}{*}\ 1\ 1\ 1 \to 1\ 1\ 1\ 1\ \underset{q_2}{1} * 1\ 1\ 1.$$

На этом заканчивается первый цикл вычитания. Теперь машина должна сравнивать числа 4 и 2. Цикл сравнения этих чисел приводит к конфигурации $11\alpha^3 * \beta^3 q_3\lambda$ или $11\alpha\alpha\alpha * \beta\beta\beta\underset{q_3}{\lambda}$, а цикл вычитания – к конфигурации $11q_2 1 * 111$ или $11\underset{q_2}{1} * 111$.

Третий цикл сравнения чисел 2 и 2 приводит к конфигурации $q_2\lambda\alpha^3 * \beta^3$ или $\underset{q_2}{\lambda}\alpha\alpha\alpha * \beta\beta\beta$, а цикл вычитания – к конфигурации $111q_4\lambda$ или $111\underset{q_4}{\lambda}$. Ставим на начало слова указатель, получаем заключительную конфигурацию $K = q_0 111$. Получили верный ответ, т. к. $НОД(4,\ 6) = 2$.

Пример 3.10. Построить в алфавите $A = \{0,1\}$ машину Тьюринга, которая применима к словам вида $1^{2m+1}01^{2n+1}$ $(m \geq 0, n \geq 0)$ и $1^{2m}01^{2n}$ $(m \geq 1, n \geq 1)$, но не применима к словам вида $1^{2m}01^{2n-1}$ и $1^{2m-1}01^{2n}$ $(m \geq 1, n \geq 1)$. (К словам иного вида машина может быть как применима, так и не применима).

Решение: как и раньше предполагаем, что q_1 – начальное состояние, q_0 – заключительное состояние, и в начальный момент указатель машины обозревает самую левую единицу на ленте. Попытаемся реализовать в «конструируемой» машине следующую идею: машина «запоминает», четным или нечетным является число единиц в первом единичном массиве слова, и

затем «сравнивает» эту характеристику с такой же характеристикой второго единичного массива.

Пара команд $q_1 1 \to q_2 1R$ и $q_2 1 \to q_1 1R$ позволяет «выяснить» четность–нечетность числа единиц в первом единичном массиве: если на «промежуточном» нуле указатель оказывается в состоянии q_1, то число единиц в первом единичном массиве четное, а если она оказывается в состоянии q_2, то число единиц в этом массиве нечетное. Проходя промежуточный нуль, нужно «запомнить», четное или нечетное число единиц было в первом массиве, с таким расчетом, чтобы после прохождения второго единичного массива в случае совпадения «четностей» числа единиц в двух массивах машина остановилась, а в ином случае не остановилась.

Потому при переходе через промежуточный нуль оба состояния q_1 и q_2 «сменим»: рассмотрим команды $q_1 0 \to q_3 0R$ и $q_2 0 \to q_4 0R$.

Второй единичный массив будем «просматривать» с помощью состояний q_3 и q_4. Если первый единичный массив «четный», то просмотр второго единичного массива будет начат в состоянии q_3, в случае, когда второй массив тоже четный, на первый нуль за этим массивом указатель «выйдет» в состоянии q_3, и машина должна будет остановиться.

Значит, можно взять команду вида $q_3 0 \to q_0 \alpha S$ ($\alpha \in \{0,1\}$ и $S \in \{R, L, E\}$) либо не включать в программу ни одной команды, начинающейся с символов q_3 и 0.

Если же первый единичный массив четный, а второй нечетный, то на первый нуль после второго массива указатель выйдет в состоянии q_4, и машина не должна остановиться. Поэтому берем команду $q_4 0 \to q_4 0E$. Аналогично рассматривается случай с нечетным единичным массивом.

Подходящая машина задается программой:

$$P:\begin{cases} q_1 1 \to q_2 1R \\ q_2 1 \to q_1 1R \\ q_1 0 \to q_3 0R \\ q_2 0 \to q_4 0R \\ q_3 1 \to q_4 1R \\ q_4 1 \to q_3 1R \\ q_4 0 \to q_4 0E \end{cases}.$$

Пример 3.11. Для функции $f(x) = 2x - 1$ построить в алфавите $A = \{0,\ 1\}$ МТ T_f, правильно вычисляющую эту функцию.

Решение: имеем $f(x) = \begin{cases} 2x - 1 & \textit{при } x \geq 1 \\ \textit{не определено} & \textit{при } x = 0 \end{cases}$.

Надо построить такую МТ T_f, которая «перерабатывает» любое слово 1^{m+1} (при $m \geq 1$) в слово 1^{2m} и удовлетворяет условию: либо $T_f(1)$ не определено, либо $T_f(1)$ не является основным машинным кодом никакого числа из N_0.

Попытаемся реализовать в «конструируемой» машине следующую идею: стирая самую левую единицу в слове 1^{m+1} ($m \geq 0$) и проходя оставшееся подслово 1^m слева направо, указатель пропускает одну (пустую) ячейку («разделительный нуль») и печатает две единицы подряд; получается слово $1^m 01^2$. Затем указатель возвращается к левой единице подслова 1^m (если такая единица есть), вычеркивает ее, проходит новое подслово 1^{m-1} слева направо, потом проходит «разделительный нуль» и две ранее напечатанные единицы (справа от него), печатает еще две единицы и опять возвращается к началу «остатка» исходного слова 1^{m+1} и т. д. Когда в исходном слове все единицы будут вычеркнуты, а во «вновь формируемом» массиве все единицы будут напечатаны, то на ленте «останется» массив из $2m + 2$ единиц; вычеркнем в нем две первые единицы и «процесс вычисления» закончим.

Программа машины Тьюринга T_f, реализующей описанную идею, выглядит так:

	q_1	q_2	q_3	q_4	q_5	q_6	q_7
0	$q_4 0R$	$q_3 0R$	$q_4 1R$	$q_5 1L$	$q_6 0L$	$q_1 0R$	–

1	$q_2 0R$	$q_2 1R$	$q_3 1R$	$q_7 0R$	$q_5 1L$	$q_6 1L$	$q_0 0R$

Имеем $T_f(1^{m+1}) = 1^{2m}$ при $m \geq 1$ и $T_f(1)$ – пустое слово. Значит, машина T_f действительно вычисляет заданную функцию $f(x)$, но не является машиной Тьюринга, правильно вычисляющей эту функцию (так как она применима к слову 1, соответствующему значению $x = 0$, а на этом значении аргумента функция $f(0)$ не определена).

Чтобы из машины T_f построить машину T_f' , правильно вычисляющую функцию $f(x)$, достаточно удалить из программы машины T_f команду $q_7 1 \to q_0 0R$ и добавить команды $q_7 1 \to q_8 0R$ и $q_8 0 \to q_8 0E$.

Задачи

3.1. Выяснить, применима ли машина Тьюринга *T*, задаваемая программой P, к слову W. Если применима, то выписать результат применения машины T к слову W. Предполагается, что q_1 – начальное состояние, q_0 – заключительное состояние, и в начальный момент указатель машины обозревает самую левую единицу на ленте.

1) P : $\begin{cases} q_1 0 \to q_1 0R \\ q_1 1 \to q_2 0R \\ q_2 1 \to q_1 0R \\ q_2 0 \to q_0 1E \end{cases}$ а) $W = 1^3 01$; б) $W = 1^2 0^2 1$; в) $W = 1^6$

2) P: $\begin{cases} q_1 0 \to q_2 1L \\ q_1 1 \to q_2 1R \\ q_2 1 \to q_1 1R \end{cases}$ а) $W = 1^2 0^2 1$; б)$W = 1^6$; в) $W = 1^2 01^3$

3) P: $\begin{cases} q_1 0 \to q_2 1R \\ q_1 1 \to q_2 1L \\ q_2 0 \to q_3 1R \\ q_2 1 \to q_3 0R \\ q_3 1 \to q_1 1R \end{cases}$ а) $W = 1^3 01^2$; б)$W = 1^5$; в) $W = 1^2 [10]^2$

4) P: $\begin{cases} q_1 0 \to q_1 1R \\ q_1 1 \to q_2 0R \\ q_2 0 \to q_1 1R \\ q_2 1 \to q_3 1L \\ q_3 0 \to q_1 1L \end{cases}$

а) $W = [10]^2 1$; б) $W = 10^2 1^2$; в) $W = 10^3 1$

5) P: $\begin{cases} q_1 0 \to q_2 1R \\ q_1 1 \to q_2 1R \\ q_2 0 \to q_3 0R \\ q_2 1 \to q_1 0L \\ q_3 0 \to q_2 1E \\ q_3 1 \to q_0 0L \end{cases}$ а) $W = 1^2$; б) $W = 1^2 0^2 1$; в) $W = 10^4 1$.

3.2. По заданной машине Тьюринга T и начальной конфигурации K_1 найти заключительную конфигурацию (q_0 – заключительное состояние):

1)T: $\begin{cases} q_1 0 \to q_2 1R \\ q_1 1 \to q_2 0L \\ q_2 1 \to q_1 1L \\ q_2 0 \to q_0 1E \end{cases}$ а) $K_1 = 1^2 01 q_1 1^2$; б) $K_1 = 101 q_1 01^2$

2)T: $\begin{cases} q_1 0 \to q_1 1L \\ q_1 1 \to q_2 1R \\ q_2 1 \to q_1 0R \\ q_2 0 \to q_0 0L \end{cases}$ а) $K_1 = 1 q_1 01^3$; б) $K_1 = 1 q_1 1^4$

3)T: $\begin{cases} q_1 0 \to q_2 0L \\ q_1 1 \to q_1 0R \\ q_2 0 \to q_2 1L \\ q_2 1 \to q_0 0R \end{cases}$ а) $K_1 = 10^3 q_1 01$; б) $K_1 = 1^2 q_1 1^3 01$

4)T : $\begin{cases} q_1 0 \to q_0 1E \\ q_1 1 \to q_2 0R \\ q_2 0 \to q_1 0R \\ q_2 1 \to q_2 1L \end{cases}$ а) $K_1 = 1^2 q_1 1^3 01$; б) $K_1 = 1 q_1 1^4$.

3.3. Построить в алфавите $\{0, 1\}$ машину Тьюринга, обладающую следующими свойствами (предполагается, что в начальный момент указатель обозревает самый левый символ слова, и в качестве пустого символа берется 0):

1) машина имеет одно состояние, одну команду и применима к любому слову в алфавите$\{0, 1\}$;

2) машина имеет две команды, не применима ни к какому слову в алфавите $\{0,\ 1\}$, и зона работы на каждом слове бесконечная;

3) машина имеет две команды, не применима ни к какому слову в алфавите $\{0,\ 1\}$, и зона работы на любом слове ограничена одним и тем же числом ячеек, не зависящим от выбранного слова;

4) машина имеет три команды, применима к словам $10^{2n}1$ $(n \geq 1)$ и не применима к словам $10^{2n+1}1$ $(n \geq 0)$;

5) машина имеет пять команд, применима к словам 1^{3n} $(n \geq 1)$ и не применима к словам $1^{3n+\alpha}$ $(\alpha = 1,\ 2$ и $n \geq 0)$;

6) машина применима к словам 1^n01^n, где $n \geq 1$, и не применима к словам 1^m01^n, где $m \neq n$, $m \geq 1$ и $n \geq 1$.

3.4. Сколько существует неэквивалентных машин Тьюринга в алфавите $\{0,\ 1\}$, программы которых содержат только по одной команде?

3.5. По программе машины Тьюринга *T* написать аналитическое выражение для функций $f(x)$ и $f(x, y)$, вычисляемых машиной *T*. (В качестве начального состояния берется q_1, а в качестве заключительного – q_0).

1) T:

	q_1	q_2
0	$q_2 1L$	$q_0 0R$
1	$q_1 1R$	$q_2 1L$

2) T:

	q_1	q_2
0	$q_2 0R$	$q_1 0L$
1	$q_1 1R$	$q_0 0R$

3) T:

	q_1	q_2	q_3
0	$q_2 1R$	$q_2 0L$	$q_0 0R$
1	$q_1 0R$	$q_3 1L$	$q_3 1R$

4) T:

	q_1	q_2	q_3
0	$q_0 0R$	$q_3 1L$	$q_1 1L$
1	$q_2 1R$	$q_1 0R$	$q_3 1R$

5) T:

	q_1	q_2	q_3	q_4
0	$q_2 0R$	$q_3 0R$	$q_0 1E$	$q_2 0R$
1	$q_2 0R$	$q_4 0R$	$q_3 1L$	$q_4 1R$

6) T:

	q_1	q_2	q_3	q_4
0	$q_2 0L$	$q_3 0L$	–	$q_1 0R$
1	$q_2 1R$	$q_2 1R$	$q_4 0L$	$q_4 1L$

3.6. Какие одноместные функции вычисляются всеми такими машинами Тьюринга в алфавите $\{0,1\}$, программы которых содержат только по одной команде?

3.7. Построить машину Тьюринга, правильно вычисляющую функцию f:

1) $f(x,\ y,\ z) = I_1^3(x,\ y,\ z) = x$; 2)$f(x,\ y,\ z) = I_2^3(x,\ y,\ z) = y$;

3)$f(x,\ y,\ z) = I_3^3(x,\ y,\ z) = z$; 4)$f(x) = 2 + x$;

5)$f(x,\ \mathrm{y}) = 2 + x$; 6)$f(x) = 2x$; 7)$f(x,\ y) = 2x + y$;

8)$f\ (x) = \begin{cases} 1, \text{ если x делится на 3} \\ 0, \text{ если не делится на 3} \end{cases}$; 9)$f(x) = \left[\frac{x}{3}\right]$;

10)$f(x) = \frac{x}{3}$; 11)$f\ (x) = x \dot{-} 1 = \begin{cases} x - 1, \text{ если } x \geq 1 \\ 0, \text{ если } x = 0 \end{cases}$;

12)$f\ (x) = x - 1 = \begin{cases} x - 1, \text{ если x} \geq 1 \\ \text{не определена, если } x = 0 \end{cases}$; 13) $f(x) = 3x$;

14) $sg\ (x) = \begin{cases} 0, \text{ если x} = 0 \\ 1, \text{ если x} \geq 1 \end{cases}$; 15) $\overline{sg}(x) = \begin{cases} 1, \text{ если } x = 0 \\ 0, \text{ если } x \geq 1 \end{cases}$;

16)$sg\ (x - 3)$; 17) $sg\ (x \dot{-} 3)$; 18) $\overline{sg}\ (x - 3)$;

19) $\overline{sg}\ (x \dot{-} 3)$; 20) $f(x) = x \dot{-} 3$; 21) $f(x) = x - 3$;

22)$f(x) = 2x - 3$; 23)$f(x) = 2x \dot{-} 3$; 24) $sg\ (2x - 3)$;

25) $sg\ (2x \dot{-} 3)$; 26) $\overline{sg}\ (2x - 3)$; 27) $\overline{sg}\ (2x \dot{-} 3)$;

28) $f(x) = x + 2$ в десятичной системе счисления;

29) $f(x) = x \dot{-} 1$ в десятичной системе счисления;

30) $f(x) = x \dot{-} 2$ в десятичной системе счисления;

31) $f\ (x, y) = x \dot{-} y = \begin{cases} x - y, & \text{если } x \geq y \\ 0, & \text{если } x < y \end{cases}$; 32)$f(x,\ y) = x - y$;

33) $f(x) = x^2$; 34) $f(x,\ y) = x \cdot y$; 35) $f(x) = |x - 5|$;

36) $f(x) = max(x, 5)$; 37)$f(x, y) = max(x, y)$;

38)$f(x) = \left[\frac{1}{x}\right] = \begin{cases} 1, & \text{если } x = 1 \\ 0, & \text{если } x \geq 2 \\ \text{не определено}, & \text{если } x = 0 \end{cases}$; 39)$f(x) = \frac{2}{4-x}$

40) $f(x) = \left[\frac{1}{x-3}\right]$; 41) $f(x) = 2^{1-x}$; 42) $f(x,\ y) = \frac{x}{2} + y$;

43) $f(x,\ y) = \frac{2+x}{2-y}$; 44) $f(x,\ y) = \frac{4-2x}{y^2}$; 45) $f(x,\ y) = \frac{2-x}{2|3-y^2|}$.

3.8. Построить МТ, которая подсчитывает количество букв «a» в любом слове, записанном в алфавите $\{\lambda,\ a,\ b\}$.

3.9. Построить МТ, которая подсчитывает количество букв «b» в слове, записанном в алфавите $A = \{\lambda,\ a,\ b\}$ и содержащем более одной буквы «b», иначе машина должна заменить исходное слово на «bbb».

3.10. Построить МТ, преобразующую любое слово в алфавите $A = \{\lambda,\ a,\ b\}$, содержащее хотя бы две буквы «a», в слово «$baabab$», иначе слово стереть.

3.11. Построить МТ, преобразующую любое слово в алфавите $A = \{\lambda,\ a,\ b\}$, содержащее хотя бы две буквы «a», в слово «$baabab$», иначе она должна оставить слово без изменения.

3.12. Построить МТ, преобразующую любое слово в алфавите$A = \{\lambda,\ a,\ b\}$, содержащее хотя бы две буквы «b», в слово «abb», иначе в слово, полученное из исходного заменой «a» на «b» и наоборот.

3.13. Построить МТ, которая любое слово в алфавите $A = \{\lambda,\ a,\ b\}$ удваивает.

3.14. Построить МТ, которая каждое слово $x_1x_2\ldots x_n$в алфавите $A = \{\lambda,\ a,\ b\}$ преобразует в слово $x_n x_{n-1}\ldots x_2 x_1$.

3.2. Операции над машинами Тьюринга

Прямое построение машин Тьюринга для решения даже простых задач может оказаться затруднительным. Однако существуют способы, позволяющие конструировать сложные машины Тьюринга из более простых. Рассмотрим основные операции над МТ: композицию (синонимы – суперпозиция, произведение), возведение в степень, ветвление и итерацию.

Всюду дальше будем рассматривать МТ, которые правильно вычисляют частичные словарные функции: $f\colon\ A^* \to A^*$.

1. Композиция (умножение)

Напомним, что композицией двух функций $f_1(R)$ и $f_2(R)$ называется функция $g(R) = f_2(f_1(R))$, которая получается при применении f_2 к результату вычисления f_1. Для того чтобы $g(R)$ была определена на данном слове R, необходимо и достаточно, чтобы f_1 была определена на R и f_2 была определена на $f_1(R)$.

Пусть машины T_1 и T_2 вычисляют соответственно $f_1(R)$ и $f_2(R)$, они имеют какие-то внешние алфавиты $A_1 = \{a_0,\ a_1,\ldots,\ a_m\}$ и $A_2 = \{a'_0,\ a'_1,\ldots,\ a'_k\}$, внутренние алфавиты $Q_1 = \{q_0,\ q_1,\ldots,\ q_n\}$ и $Q_2 = \{q'_0,\ q'_1,\ldots,\ q'_t\}$ (можно переобозначить состояния в Q_2 так, чтобы они отличались от состояний Q_1) и соответственно программы P_1 и P_2.

Определение 3.10. *Композицией (произведением)* машин $T_1 = (A_1,\ Q_1,\ P_1)$ и $T_2 = (A_2,\ Q_2,\ P_2)$ будем называть машину $T = T_1 \circ T_2$ с внешним алфавитом $A_1 \cup A_2$, внутренним алфавитом $Q = \{q_1,\ldots,\ q_n,\ q_0 = q'_1,\ q'_2,\ldots,\ q'_t\}$ и программой $P = P'_1 \cup P'_2$, где P'_1 получается из P_1 заменой во всех командах программы P_1 заключительного символа q_0 на символ q'_1 – какое-либо начальное состояние T_2, а программа P'_2 получается из P_2 следующим образом: заключительное состояние не меняется, но зато каждый из остальных символов q'_j $(j = 1,\ 2,\ldots,\ t)$ заменяем символом q_{n+j}. Новая программа P определяет машину T, называемую композицией машин T_1 и T_2

(по паре состояний $(q_0,\ q_1')$) и обозначаемую через $T_1 \circ T_2$ или T_1T_2, более подробно $T = T\,(T_1,T_2,(q_0,\ q_1'))$.

Схематически композиция обозначается следующим образом:$R \xrightarrow[T_1]{} f_1(R) \xrightarrow[T_2]{} f_2(f_1(R))$.

Замечание 1. В том случае, когда одна из перемножаемых машин имеет два или несколько заключительных состояний, умножение определяется совершенно аналогично, но только обязательно должно присутствовать указание, какое из заключительных состояний предыдущего сомножителя отождествляется с начальным состоянием последующего.

Замечание 2. Операция композиции машин ассоциативна, т. е. $(T_1 \circ T_2) \circ T_3 = T_1 \circ (T_2 \circ T_3)$, но не коммутативна $T_1 \circ T_2 \neq T_2 \circ T_1$.

Пример 3.12. Пусть машина T_1 задана тьюринговой функциональной схемой:

T_1	0	1
q_1	$q_2 0R$	$q_1 1R$
q_2	$q_2 0R$	$q_0 1E$

Эта машина отыскивает ближайшую справа группу единиц после данной группы единиц и нулей.

Машина T_2 задана таблицей:

T_2	0	1
q_1^2	$q_0^2 0E$	$q_1^2 0R$

Эта машина стирает все единицы, если они есть, до ближайшего справа нуля.

В исходном состоянии считывающие указатели машин T_1 и T_2 находятся у крайней левой заполненной ячейки.

Тогда машина $T = T\ (T_1,\ T_2,\ (q_0,\ q_1^2))$, являющаяся их произведением, будет иметь таблицу с тремя состояниями, $q_0 = q_0^2$ заключительное состояние машины T_2:

$T = T_1 \circ T_2$	0	1
q_1	$q_2 0R$	$q_1 1R$
q_2	$q_2 0R$	$q_3 1E$
q_3	$q_0 0E$	$q_3 0R$

Машина T отыскивает ближайшую справа группу единиц и стирает их.

Пример 3.13. Сконструируем машину Тьюринга, правильно вычисляющую функцию $I_2^3(x, y, z) = y$, применяя к начальной конфигурации $K = q_1 01^{x+1} 01^{y+1} 01^{z+1} 0$ композицию следующих машин МТ: Б$^+$ «правый сдвиг»,Б$^-$ «левый сдвиг», О «удаление слова», B «транспозиция».

Решение: $K = q_1 01^{x+1} 01^{y+1} 01^{z+1} 0 \xrightarrow[\text{Б}^+]{} 01^{x+1} q 01^{y+1} 01^{z+1} 0 \xrightarrow[B]{}$

$\xrightarrow[\text{В}]{} 01^{y+1} q 01^{x+1} 01^{z+1} 0 \xrightarrow[\text{Б}^+]{} 01^{y+1} 01^{x+1} q 01^{z+1} 0 \xrightarrow[O]{}$

$\xrightarrow[\text{О}]{} 01^{y+1} 01^{x+1} q 00^{z+1} 0 \xrightarrow[\text{Б}^-]{} 01^{y+1} q 01^{x+1} 00^{z+1} 0 \xrightarrow[\text{О}]{}$

$\xrightarrow[\text{О}]{} 01^{y+1} q 00^{x+1} 00^{z+1} 0 \xrightarrow[\text{Б}^-]{} q_0 01^{y+1} 00^{x+1} 00^{z+1} 0.$

Получили, что заключительной конфигурацией является $K = q_0 01^{y+1}$.

Таким образом, функция $I_2^3(x,\ y,\ z) = y$ вычисляется следующей композицией машин: Б$^+$ ^{B} Б$^+$ ^{O} Б$^-$ ^{O} Б$^-$ = Б$^+$В $^{Б^+}(ОБ^-)^2$.

2. Операция возведения в степень

n-й степенью машины Т называется произведение $TT \ldots T = T^n$ с n сомножителями.

Так, например, зная МТ T (см. пример 3.5), вычисляющую функцию $s(x) = x + 1$, можно построить МТ, вычисляющую функцию $f(x) = x + 5$. ($T \circ \text{Т} \circ \text{Т} \circ \text{Т} \circ \text{Т} = T^5$)

3. Операция итерации (или организация цикла)

Операция итерации применима к одной машине. Суть операции состоит в следующем: пусть машина T_1 имеет несколько заключительных состояний и q'– одно из них, а q'' – какое-либо состояние машины T_1, не являющееся заключительным. Заменим всюду в программе P машины T_1 символ q' на

символ q''. Получим программу $P_{\text{нов}}$, определяющую машину T (q', q''). Машина T называется *итерацией* машины T_1 по паре состояний (q', q'').

Замечание. Если T_1 имеет только одно заключительное состояние, то после выполнения итерации получается машина, не имеющая заключительного состояния вовсе.

Пример 3.14. Пусть МТ, которая, отправляясь от числа, воспринятого в стандартном положении, стирает число правее данного (если такое имеется), задана тьюринговой функциональной схемой:

	q_1	q_2	q_3	q_4	q_5	q_6
0	$q_2 0R$	$q_3 0L$	$q_4 0L$	$q_4 0L$	$q_0 0R$	$q_0 0E$
1	$q_1 1R$	$q_6 0R$	–	$q_5 1L$	$q_5 1L$	$q_6 0R$

Эта программа содержит две команды с заключительным состоянием q_0. Применив операцию зацикливания (заменим в команде $q_6 0 \to q_0 0E$ состояние q_0 состоянием q_1), получим МТ, которая, отправляясь от числа, воспринятого в стандартном положении, стирает все числа правее данного (если таковые имеются) до первого встречного промежутка из двух или более пустых ячеек и возвращается к стандартному положению первоначально воспринятого числа.

4. Ветвление

Пусть машины Тьюринга T_1, T_2и T_3 имеют общий алфавит A и задаются программами P_1, P_2 и P_3 соответственно. Считаем, что внутренние алфавиты этих машин попарно не пересекаются (иначе их внутренние состояния можно переобозначить). Пусть q_0', q_0'' – какие-либо различные заключительные состояния машины T_1. Заменим всюду в программе P_1 состояние q_0' некоторым начальным состоянием q_1^2 машины T_2, а состояние q_0'' некоторым начальным состоянием q_1^3 машины T_3. Затем новую программу объединим с программами P_2 и P_3. Получим программу P, задающую машину Тьюринга $T = T\ (T_1, (q_0', q_1^2), T_2, (q_0'', q_1^3), T_3)$.

Эта машина называется *разветвлением* машин T_2 и T_3, управляемым машиной T_1.

Пример 3.*15.* $sg\ (x) = \begin{cases} 0, & если \quad x = 0 \\ 1, & если \quad x \geq 1 \end{cases}$

	0	1	Комментарии
q_1	–	$q_2 1R$	T_1, если $f(0) = 0$
q_2	$q_0 0L$	$q_3 1R$	
q_3	$q_4 0L$	$q_3 0R$	T_2 стирает слово
q_4	$q_4 0L$	$q_5 1L$	T_3 возврат на начало
q_5	$q_0 0R$	$q_5 1L$	

$\mathrm{T} = \mathrm{T}_1 \vee \mathrm{T}_2 \circ \mathrm{T}_3$, здесь T_1 вычисляет значение $sg\ (x)$ для $x = 0$, а $\mathrm{T}_2 \circ \mathrm{T}_3$ вычисляет значение $sg\ (x)$ для $x \geq 1$.

Вывод: Машина Тьюринга дает один из путей уточнения понятия алгоритма. В связи с этим возникают вопросы: можно ли считать, что способ задания алгоритмов с помощью машины Тьюринга является универсальным? Может ли всякий алгоритм задаваться таким образом? На эти вопросы современная теория алгоритмов предлагает ответ в виде следующей гипотезы, которая называется тезисом Тьюринга.

Тезис Тьюринга: всякий алгоритм может быть реализован машиной Тьюринга.

Это утверждение нельзя доказать, так как оно связывает нестрогое интуитивное понятие алгоритма со строгим определением понятия машины Тьюринга. Эта гипотеза может быть опровергнута, если удастся привести пример алгоритма, который не может быть реализован с помощью тьюринговой функциональной схемы. Однако все известные до сих пор алгоритмы могут быть реализованы в виде машины Тьюринга.

Задачи

1. Построить композицию T_1T_2 машин Тьюринга T_1 и T_2 по паре состояний $(q_{10},\ q_{21})$ и найти результат применения композиции T_1T_2 к слову R (q_{20} – заключительное состояние машины T_2):

1)

T_1:

	0	1
q_{11}	$q_{12}0R$	$q_{12}1R$
q_{12}	$q_{10}1L$	$q_{11}0R$

T_2:

	0	1
q_{21}	$q_{22}1R$	$q_{21}0L$
q_{22}	$q_{21}1R$	$q_{20}1E$

а)$R = 1^3 0^2 1^2$; б)$R = 1^4 01\ [110]^3$; в)$R = 1^4 0^2 1^3 01^2$; г)$R = 1^2 [01]^2 1^2$; д)$R = [10]^3 [01]^2 1^2$.

2)

T_1:

	0	1
q_{11}	$q_{10}0L$	$q_{12}1R$
q_{12}	$q_{13}0R$	$q_{13}1R$
q_{13}	$q_{11}0R$	$q_{11}0R$

T_2:

	0	1
q_{21}	$q_{22}0L$	$q_{21}1L$
q_{22}	$q_{23}0L$	$q_{22}1L$
q_{23}	$q_{20}0R$	$q_{23}1L$

а) $R = 1^3 0^2 1^2$; б)$R = 1^4 01\ [110]^3$; в)$R = 1^4 0^2 1^3 01^2$;

г)$R = 1^2 [01]^2 1^2$; д)$R = [10]^3 [01]^2 1^2$.

T_1:

	0	1
q_{11}	$q_{12}0R$	$q_{11}1R$
q_{12}	$q_{13}0R$	$q_{11}1R$
q_{13}	$q_{10}1L$	–

T_2:

	0	1
q_{21}	$q_{22}0L$	$q_{21}1L$
q_{22}	$q_{23}0L$	$q_{22}1L$
q_{23}	$q_{20}0R$	$q_{23}1L$

а) $R = 1^3 0^2 1^2$; б)$R = 1^4 01\ [110]^3$; в)$R = 1^4 0^2 1^3 01^2$;

г)$R = 1^2 [01]^2 1^2$; д)$R = [10]^3 [01]^2 1^2$.

2. Найти результат применения итерации машины T по паре состояний (q_0, q_i) к слову R (заключительными состояниями являются q_0 и q_0').

1) i=1, T:

	q_1	q_2	q_3	q_4	q_5
0	$q_0$0E	$q_4$0E	$q_5$0E	$q_4$1R	q_0'1L
1	$q_2$0R	$q_3$0R	$q_1$0R	–	–

а) $R = 111101$; б) $R = 1110111$; в) $R = 1^{3к}$;

г) $R = 1^{3к+1}$; д) $R = 1^{3к+2}$ (к ≥1).

2) i=1, T:

	q_1	q_2	q_3	q_4	q_5	q_6
0	q_0'0R	q_0'0R	$q_4$0R	$q_5$1L	$q_6$0L	$q_0$0R
1	$q_2$0R	$q_3$0L	$q_3$1R	$q_4$1R	$q_5$1L	$q_6$1L

а) $R = 111101$; б) $R = 1110111$; в) $R = 1^4 0^2 1^3 01^2$;

г) $R = 1^{2x}$; д) $R = 1^{2x+1}$ $(x \geq 1)$.

3) i = 3; T:

	q_1	q_2	q_3	q_4	q_5	q_6
0	$q_2$0L	q_0'1E	$q_4$0R	$q_5$1L	$q_6$0L	$q_2$0R
1	$q_1$2R	$q_2$1R	$q_3$1R	$q_4$1R	$q_5$1L	$q_6$1L
2	–	$q_3$1R	–	–	–	$q_0$1R

а) $R = 1^4 0^3 1^2$; б)$R = 101\ [110]^2$; в) $R = 1^3 0^4 1^2 01$;

г) $R = 1^2[01]^2 1^2$; д) $R = 1^x 01^y$ ($x \geq 1, y \geq 1$).

3. Найти результат применения итерации машины

$T = T$ (T_1, (q'_{10}, q_{21}), T_2, (q''_{10}, q_{31}), T_3) к слову R (q_{20}– заключительное состояние T_2, а q_{30}– заключительное состояние T_3):

T_1:

	q_{11}	q_{12}
0	$q_{12}0R$	$q'_{10}0R$
1	$q_{12}1R$	$q''_{10}1L$

T_2:

	q_{21}
0	$q_{20}1E$
1	$q_{21}0R$

T_3:

	q_{31}	q_{32}
0	$q_{32}1L$	$q_{30}1E$
1	$q_{31}1R$	$q_{32}0L$

а) $R = 101^3$; б)$R = 1^3 01$; в) $R = 1^x 0^2 1$ $(x \geq 1)$;

г) $R = 1^x 0101^y 01^z$ ($x \geq 1$, $y \geq 1, z \geq 1$).

2)

T_1:

	q_{11}	q_{12}	q_{13}
0	$q_{12}0R$	$q'_{10}0L$	$q''_{10}0R$
1	$q_{11}1R$	$q_{13}1R$	$q_{13}1R$

T_2:

	q_{21}	q_{22}
0	$q_{22}0L$	$q_{20}1R$
1	$q_{21}1L$	$q_{22}0L$

T_3:

	q_{31}	q_{32}
0	$q_{32}0R$	$q_{30}1E$
1	$q_{31}1R$	$q_{31}1R$

а) $R = 101^3$; б)$R = 1^3 01$; в) $R = 1^x 0^2 1$ $(x \geq 1)$;

г) $R = 1^x 0101^y 01^z$ ($x \geq 1$, $y \geq 1, z \geq 1$).

4. Постройте машину Тьюринга, которая называется «перенос нуля» и обозначается А, осуществляющую перевод начальной конфигурации $q_1 001^x 0$ в заключительную конфигурацию $q_0 01^x 00$.

5. Постройте машину Тьюринга, которая называется «левый сдвиг» и обозначается Б$^-$, осуществляющую перевод начальной конфигурации $01^x q_1 0$ в заключительную конфигурацию $q_0 01^x 0$.

6. Постройте машину Тьюринга, которая называется «правый сдвиг» и обозначается $Б^{+}$, осуществляющую перевод начальной конфигурации $q_1 01^{x} 0$ в заключительную конфигурацию $01^{x} q_0 0$.

7. Постройте машину Тьюринга, называемую «транспозиция» и обозначаемую В, которая перерабатывает начальную конфигурацию $01^{x} q_1 01^{y} 0$ в заключительную конфигурацию $01^{y} q_0 01^{x} 0$.

8. Постройте машину Тьюринга, называемую «удвоение» и обозначаемую Г, которая перерабатывает начальную конфигурацию $q_1 01^{x} 0$ в заключительную конфигурацию $q_0 01^{x} 01^{x} 0$.

9. Постройте машину Тьюринга, называемую «циклический сдвиг» и обозначаемую Ц, которая перерабатывает начальную конфигурацию $01^{x} 01^{y} q_1 01^{z} 0$ в заключительную конфигурацию $01^{z} q_0 01^{x} 01^{y} 0$.

10. Постройте машину Тьюринга, называемую «копирование» и обозначаемую $К_2$, которая перерабатывает начальную конфигурацию $q_1 01^{x} 01^{y}$ в заключительную конфигурацию $01^{x} 01^{y} q_0 01^{x} 01^{y}$.

Указание. Проверьте, что эта машина представляет собой следующую композицию построенных выше машин: $К_2 = Б^{+} ГВБ^{+} ВГВБ^{+} ВБ^{-} Б^{-} ВБ^{+}$.

Раздел 4. Функции, вычислимые по Маркову

4.1. Нормальный алгоритм Маркова

Теория нормальных алгоритмов (или алгорифмов – в авторской транскрипции) была разработана советским математиком А.А.Марковым (1903–1979) в конце 40-х годов XX в. Данный подход связывает неформальное понятие алгоритмической вычислимости с переработкой слов в некотором конечном алфавите в соответствии с определенными правилами. В качестве элементарного преобразования используется подстановка одного слова вместо другого. Множество таких подстановок определяет схему алгоритма. Правила, определяющие порядок применения подстановок, а также правила останова являются общими для всех нормальных алгоритмов.

Дадим формальные определения. *Алфавитом* будем называть всякое непустое конечное множество символов, не содержащее знаки $\rightarrow$ и $\cdot$. Его элементы будем называть *буквами*, а любые последовательности букв – *словами* в алфавите A. Пустую последовательность букв будем называть *пустым словом* и обозначать λ. Если A и B – два алфавита, причем $A \subseteq B$, то алфавит B будем называть *расширением* алфавита A.

Определим понятие вхождения слова P в данное слово R.

Определение 4.1. Будем говорить, что имеется вхождение слова P в слово R (или слово P входит в R), если существуют слова V_1 и V_2 (возможно, пустые) такие, что $R = V_1PV_2$ (1). Если слово V_1 имеет наименьшую длину из всех слов вида (1), то говорят о первом вхождении P в слово R.

Пример 4.1. В алфавите $A=\{a,b\}$ рассмотрим следующие слова: 1) $P_1=aa$ входит в $R=abaabab$, так как $R=V_1P_1V_2$, где $V_1=ab$, $V_2=bab$. В этом случае V_1 и V_2 определены однозначно.

2) $P_2=aba$ входит в $R = abaabab$, так как $R=V_1P_2V_2$, где $V_1=\lambda$, $V_2=abab$; и $R = W_1P_2W_2$, где $W_1=aba$, $W_2=b$. В этом случае V_1 и V_2 определены неоднозначно.

З*амечание*. В дальнейшем будем разыскивать первые вхождения данных слов в некоторые слова и заменять их на другие слова, возможно, пустые.

Пример 4.2. В результате замены первого вхождения слова $P=ba$ в слово $R=aababab$ на слово 1) $Q=a$; 2) $Q=b$; 3) $Q=baa$; 4) на пустое слово получим следующие слова:1) *aaabab*; 2) *aabbab*; 3) *aabaabab*; 4)*aabab*.

Если же первое вхождение пустого слова в любое слово R заменить на некоторое слово Q, получится слово QR, т. е. справа к слову Q дописывается слово R.

Например, если $Q=aba$, $R=abbb$, то $QR=abaabbb$.

Пусть A – алфавит, не содержащий в качестве букв знаки $\rightarrow$ и $\cdot$. *Обычной формулой подстановки* в алфавите A называется выражение $P\rightarrow Q$, где P и Q – слова в алфавите A. *Заключительной формулой подстановки* в алфавите A называется выражение $P\rightarrow \cdot Q$, где P и Q – слова в алфавите A. P и Q называются левой и правой частью формулы подстановки.

Определение 4.2. *Марковской подстановкой* ($P\rightarrow Q$ или $P\rightarrow \cdot Q$) называют операцию над словами в данном алфавите A. Эта операция задается с помощью упорядоченной пары слов $(P,\ Q)$ и состоит в том, что в заданном слове R находят *первое* вхождение слова P (если таковое имеется) и, не изменяя остальных частей слова R, заменяют в нем это вхождение словом Q. Полученное слово называется результатом применения марковской подстановки к слову R. Если же первого вхождения слова P в слове R нет (и, следовательно, нет вообще ни одного вхождения P в R), то считается, что марковская подстановка не применима к слову R.

Пример 4.3. Пусть для слов в алфавите $A=\{a,b,c\}$ заданы следующие марковские подстановки: а) $ca\rightarrow ab$; б) $bca\rightarrow\lambda$; в) $abca\rightarrow a$; г) $b\rightarrow a$.

Нужно применить каждую из них к слову $abcabcabcabcab$.

Решение: а) Последовательность получающихся слов можно представить следующим образом (происходит последовательная замена в данном слове всех вхождений букв caна буквы ab):

$$ab\underline{ca}bcabcabcab \to ababb\underline{ca}bcabcab \to ababbabb\underline{ca}bcab \to$$
$$\to ababbabb\underline{ca}bcab \to ababbabbabb\underline{ca}b \to ababbabbabbabb$$

б) $a\underline{bca}bcabcabcab \to a\underline{bca}bcabcab \to a\underline{bca}bcab \to a\underline{bca}b \to ab$;

в) $\underline{abca}bcabcabcab \to \underline{abca}bcabcab \to \underline{abca}bcab \to \underline{abca}b \to \to ab$;

г)$a\underline{b}cabcabcabcab \to aaca\underline{b}cabcabcab \to aacaaca\underline{b}cabcab \to\to$
$aacaacaaca\underline{b}cab \to aacaacaacaaca\underline{b} \to aacaacaacaacaa.$

Определение 4.3. Произвольная конечная не пустая упорядоченная последовательность марковских подстановок называется *нормальной схемой в алфавите A* и обозначается S_a. Таким образом, схема нормального алгоритма имеет вид:

$$S_a: \quad \begin{cases} P_1 \to (\cdot)Q_1 \\ P_2 \to (\cdot)Q_2 \\ \quad \dots \\ P_m \to (\cdot)Q_m \end{cases}, \text{ здесь } P_i,\ Q_i \in A \text{ или } P_i,\ Q_i \in B \supseteq A$$

Запись $P\to(\cdot)Q$ обозначает одну из формул подстановок $P\to Q$ или $P\to \cdot\, Q$.

Если схема нормального алгоритма задана в некотором расширении B алфавита A ($A\subseteq B$), то говорят, что задана нормальная схема *над A* или нормальная схема *в алфавите B.*

Нормальный алгоритм (н. а.) над алфавитом A задается конечным алфавитом B, не содержащим $\to$ и $\cdot$ и включающим в себя A или совпадающим с A (т. е. $B \supseteq A$), и нормальной схемой в алфавите B.

Определение 4.4. *Нормальным алгоритмом (Маркова) над алфавитом A* называется следующее правило построения последовательности $R \Rightarrow R_1 \Rightarrow \cdots \Rightarrow R_i \Rightarrow R_{i+1} \Rightarrow \cdots$ слов в алфавите $B\supseteq A$, исходя из данного слова R в алфавите A. Пусть для некоторого $i \geq 0$ слово R_i построено и процесс построения рассматриваемой последовательности еще не завершился. Если при этом в схеме нормального алгоритма нет формул, левые части которых входили бы в

R_i, то R_{i+1} полагают равным R_i, и процесс построения последовательности считается завершившимся. Если же в схеме имеются формулы с левыми частями, входящими в R_i, то в качестве R_{i+1} берется результат марковской подстановки правой части первой из таких формул вместо первого вхождения ее левой части в слово R_i; процесс построения последовательности считается завершившимся, если на данном шаге была применена формула заключительной подстановки, и продолжающимся – в противном случае.

Если процесс построения последовательности слов обрывается $R \Rightarrow R_1 \Rightarrow \cdots \Rightarrow R_{m-1} \Rightarrow R_m$, то говорят, что рассматриваемый нормальный алгоритм *применим к слову* R. Последний член последовательности R_m называется *результатом* применения нормального алгоритма *к слову* R. Говорят, что нормальный алгоритм *перерабатывает* R в R_m.

Если последовательность слов бесконечна, то говорят, что нормальный алгоритм *не применим к слову* R.

Пример 4.4. Тождественный нормальный алгоритм над алфавитом A применим к каждому слову в алфавите A, и результатом его работы является это же слово.

Такой н. а. может быть задан алфавитом $B = A$ (не содержащим $\rightarrow$ и $\cdot$) и нормальной схемой $\{\rightarrow\cdot$, т. е. в этой подстановке имеется только одна формула подстановки, являющаяся заключительной, с пустыми левыми и правыми частями.

Замечание. Рассмотренный н. а. вычисляет тождественную всюду определенную словарную функцию $F(R) = R$ или числовую функцию $f(x) = x$.

Пример 4.5. Нормальный алгоритм над алфавитом $A = \{a,\ b\}$, заданный алфавитом $B = A$ и нормальной схемой S_a: $\begin{cases} a \rightarrow b \\ b \rightarrow a \end{cases}$ не применим ни к одному слову в алфавите A.

Замечание. Нормальная схема S_a: $\{ \rightarrow$ вычисляет нигде не определенную функцию, так как эта схема вообще не применима ни к одному слову.

Пример 4.6. Нормальный алгоритм, распознающий слова в алфавите $A = \{a,\ b\}$, содержащие равное число букв *a* и *b*.

Будем считать, что этот алгоритм перерабатывает слова, содержащие равное число букв *a* и *b* (и только их), в пустое слово, а другие слова оставляет без изменения. В таком случае искомый алгоритм можно представить в виде следующего предписания:

1. Подставь в данное слово вместо первого вхождения слова *ba* слово *ab*. Переходи к указанию 2.
2. Если во вновь получившемся слове есть вхождение слова *ba*, то переходи к указанию 1, иначе – к указанию 3.
3. Выброси первое вхождение слова *ab*, переходи к указанию 4.
4. Если в получившемся слове есть вхождение слова *ab*, то переходи к указанию 3, иначе – к указанию 5.
5. Сравни полученное слово с пустым словом. Если оно совпадает с ним, то переходи к указанию 6, иначе к указанию 7.
6. Исходное слово содержит равное число букв *a* и *b*. Переходи к указанию 8.
7. Исходное слово содержит неравное число вхождений букв *a* и *b*. Переходи к указанию 8.
8. Процесс вычисления остановить.

В основе этого алгоритма заложен следующий принцип: в перерабатываемом слове *P* в алфавите $A = \{a,\ b\}$ делается перестановка букв *a* и *b*, пока слово не примет вид: $aa\ldots abb\ldots b$ (все буквы *a* стоят левее всех букв *b*). В получившемся слове затем последовательно уничтожается равное число букв *a* и *b* (каждое слово *ab* заменяется пустым словом).

Этот алгоритм можно задать алфавитом $B = A = \{a,\ b\}$ и нормальной схемой S_a: $\begin{cases} ba \to ab \\ ab \to \end{cases}$.

Работа этого н. а., например, над словом $R = ababba$, состоит из следующей последовательности слов:

$ab\underline{ab}ba \to aabb\underline{ba} \to aab\underline{ba}b \to aa\underline{ba}bb \to aa\underline{ab}bb \to a\underline{ab}b \to \underline{ab} \to \lambda.$

Поскольку слово $R = ababba$ переработано алгоритмом в пустое слово λ, получаем, что оно содержит равное число букв *a* и *b*.

Пример 4.7. Нормальный алгоритм над алфавитом *A* левого присоединения фиксированного слова *Q* применим к каждому слову в алфавите *A*, и результатом его работы над словом *R* является слово *QR*.

Такой н. а. может быть задан алфавитом $B = A$ (не содержащим $\to$ и $\cdot$) и нормальной схемой $\{\to\cdot\ Q$, т. е. в этой подстановке имеется только одна формула подстановки, являющаяся заключительной, с пустой левой частью.

Пример 4.8. Нормальный алгоритм над алфавитом $A = \{a,\ b\}$ правого присоединения фиксированного слова *Q* применим к каждому слову в алфавите *A*, и результатом его работы над словом *R* является слово *RQ*.

Такой н. а. может быть задан алфавитом *B*=$\{a, b, c\}$ и нормальной схемой

$$S_a: \begin{cases} ca \to ac \\ cb \to bc \\ c \to\cdot\ Q \\ \to c \end{cases}$$

Работа этого н. а. над словом $R = abaa$ состоит из следующей последовательности слов:

$abaa \to \underline{ca}baa \to a\underline{cb}aa \to ab\underline{ca}a \to aba\underline{ca} \to abaa\underline{c} \to abaaQ.$

Здесь после первого шага было получено слово $cabaa$ (как результат применения последней подстановки нормальной схемы), каждый следующий шаг переработки будет сдвигать символ *c* на одно место вправо, и после пятого шага мы будем иметь слово $abaac$, которое после применения заключительной формулы подстановки даст слово $abaaQ = RQ$.

Замечание. Рассмотренный нормальный алгоритм вычисляет словарную функцию $F(R) = RQ$ в алфавите $A = \{a,\ b\}$.

Пример 4.9. Нормальный алгоритм удвоения над *A* преобразует каждое слово *R* в алфавите *A* в слово *RR*.

Пусть $A = \{a,\ b\}$. Зададим н. а. удвоения над A алфавитом B=$\{a, b,\alpha,\beta\ \}$ и нормальной схемой

$$S_a: \begin{cases} \alpha a \to a\beta a\alpha \\ \alpha b \to b\beta b\alpha \\ \beta aa \to a\beta a \\ \beta ab \to b\beta a \\ \beta ba \to a\beta b \\ \beta bb \to b\beta b \\ \beta \to \\ \alpha \to \cdot \\ \to \alpha \end{cases}$$

Работа этого нормального алгоритма над словом $R = bab$ состоит из следующей последовательности слов:

$$bab \to \underline{\alpha b}ab \to b\beta b\underline{\alpha a}b \to b\beta ba\beta a\underline{\alpha b} \to b\underline{\beta ba\beta}ab\beta b\alpha \to$$

$$\to ba\beta b\underline{\beta ab}\beta b\alpha \to ba\underline{\beta bb}\beta a\beta b\alpha \to bab\underline{\beta}b\beta a\beta b\alpha \to\to babb\beta a\beta b\alpha \to$$

$$babba\underline{\beta b}\alpha \to babbab\alpha \to babbab$$ – результат.

Пример 4.10. Нормальный алгоритм обращения каждого слова в алфавите $A = \{a_1, a_2, \dots, a_n\}$ преобразует слово $R = a_{i_1} \dots a_{i_k}$ в слово $R^{-1} = a_{i_k} \dots a_{i_1}$.

Рассмотрим алфавит $B = A \cup \{\beta, \gamma\}$ (здесь β, γ – новые буквы) и соответственно схему S_a:

$$\begin{cases} 1.\quad \gamma\gamma \to \beta \\ 2.\quad \beta x \to x\beta \quad ,\forall x \in A \\ 3.\quad \beta\gamma \to \beta \\ 4.\quad \beta \to \cdot \\ 5.\quad \gamma xy \to y\gamma x,\ \forall x, y \in A \\ 6.\quad \to \gamma \end{cases}$$

Покажем, что данный алгоритм осуществляет обращение слов в алфавите A.

Доказательство. Пусть $R = a_{i_1} \dots a_{i_k}$ слово в алфавите A. Тогда

$$R \xrightarrow[(6)]{} \gamma R \xrightarrow[(5)]{} a_{i_2}\gamma a_{i_1} \dots a_{i_k} \xrightarrow[(5)]{} a_{i_2}a_{i_3}\gamma a_{i_1}a_{i_4} \dots a_{i_k} \to$$

$$\xrightarrow[(5)]{} a_{i_2}a_{i_3} \dots a_{i_k}\gamma a_{i_1} \xrightarrow[(6)]{} \gamma a_{i_2} \dots a_{i_k}\gamma a_{i_1} \to \dots \to a_{i_3}a_{i_4} \dots a_{i_k}\gamma a_{i_2}\gamma a_{i_1}.$$

Теперь, повторяя этот процесс, получим:

$$\gamma a_{i_k}\gamma a_{i_{k-1}}\gamma \dots \gamma a_{i_2}\gamma a_{i_1} \xrightarrow[(6)]{} \gamma\gamma a_{i_k}\gamma a_{i_{k-1}} \dots \gamma a_{i_2}\gamma a_{i_1} \rightarrow$$

$$\xrightarrow[(1)]{} \beta a_{i_k}\gamma a_{i_{k-1}} \dots \gamma a_{i_2}\gamma a_{i_1} \xrightarrow[(2)]{} a_{i_k}\beta\gamma a_{i_{k-1}}\gamma \dots \gamma a_{i_1} \rightarrow$$

$$\xrightarrow[(3)]{} a_{i_k}\beta a_{i_{k-1}} \dots \lambda a_{i_1} \rightarrow \cdots \xrightarrow[(2,3.4)]{} \dots a_{i_k}a_{i_{k-1}} \dots a_{i_1}.$$

Определение 4.5. Функция *f*, заданная на некотором множестве слов алфавита *A*, называется *нормально вычислимой* (или *вычислимой по Маркову*), если найдется такое расширение B данного алфавита A ($A \subseteq B$) и такая схема нормального алгоритма S_a в алфавите *B*, что каждое слово R (в алфавите *A*) из области определения функции f этот алгоритм перерабатывает в слово $f(R)$.

Класс функций, вычислимых по Маркову, обозначают через *M*.

В качестве функций *f* можно брать как словарные, так и числовые функции. Будем рассматривать числовые функции *f*, определенные на множестве N_0, принимающие значения из N_0.

Как и раньше целое неотрицательное число *x* будем изображать словом из *x*+1 единиц. Набор чисел $x_1, x_2, \dots, x_n$ условимся изображать словом $1^{x_1+1} * 1^{x_2+1} * \dots * 1^{x_n+1}$.

Пример 4.11. Нормальный алгоритм, вычисляющий функцию $o(x) = 0$, задается алфавитом $B = \{1\}$ и нормальной схемой $\{11 \rightarrow 1$.

Этот алгоритм может быть сформулирован в виде следующего словесного предписания: «Сотри у данного слова (последовательности единиц) все единицы, кроме одной (соответствует нулю), и остановись. Полученное слово и есть результат».

Действие стирания единицы можно задать с помощью подстановки $\{11 \rightarrow 1$. Алгоритм, заданный одной этой подстановкой, будет стирать по одной единице у исходного слова, пока не останется одна последняя единица, тогда процесс закончится, так как левая часть «11» формулы подстановки не входит в слово «1». Указанный н. а. применим к каждому слову в алфавите {1},

и его работа при вычислении $o(x) = 0$ состоит из следующей последовательности слов: $1^{x+1} \rightarrow\ 1^{x} \rightarrow\ 1^{x-1} \rightarrow 1^{x-2} \rightarrow \cdots \rightarrow 1$.

Пример 4.12. Нормальный алгоритм, вычисляющий функцию $s(x) = x + 1$, задается алфавитом $B = \{1\}$ и нормальной схемой $\{1 \rightarrow\ \cdot 11$.

Рассматриваемый алгоритм – это алгоритм перехода от произвольного числа к числу на единицу больше, или алгоритм, перерабатывающий слово, изображающее число x в слово, изображающее $x + 1$. Этот алгоритм может быть сформулирован в виде следующего словесного предписания: «Припиши к данному слову (последовательности единиц) еще одну единицу и остановись. Полученное слово и есть результат».

Действие приписывания единицы можно задать с помощью подстановки $\{1 \rightarrow 11$. Но алгоритм, заданный одной этой подстановкой, будет бесконечно прибавлять по одной единице к исходному слову, никогда не останавливаясь, т.е. процесс никогда не закончится. Для указания остановки пользуются заключительной подстановкой $\{1 \rightarrow\ \cdot 11$ (точка за стрелкой указывает на то, что эта подстановка применяется только раз, после чего наступает окончание процесса). Рассматриваемый н. а. применим к каждому слову в алфавите $\{1\}$ и его работа состоит из двух слов $1^{x+1} \rightarrow\ 1^{x+2}$.

Пример 4.13. Нормальный алгоритм, вычисляющий функцию $s(x) = x + 1$ не в унарной системе (как сделано в предыдущем примере), а в десятичной системе счисления.

Решение: в качестве алфавита для записи значений аргумента x и функции $s(x)$ возьмем алфавит $A = \{0,\ 1,\ 2,\ 3, \ldots,\ 9\}$, а нормальный алгоритм будем строить в его расширении $B = A \cup \{a,\ b\}$. Схема нормального алгоритма представляет собой следующий упорядоченный набор подстановок (читается по столбцам):

$0b \to\cdot\ 1$	$a0 \to 0a$	$0a \to 0b$
$1b \to\cdot\ 2$	$a1 \to 1a$	$1a \to 1b$
$2b \to\cdot\ 3$	$a2 \to 2a$	$2a \to 2b$
$3b \to\cdot\ 4$	$a3 \to 3a$	$3a \to 3b$
$4b \to\cdot\ 5$	$a4 \to 4a$	$4a \to 4b$
$5b \to\cdot\ 6$	$a5 \to 5a$	$5a \to 5b$
$6b \to\cdot\ 7$	$a6 \to 6a$	$6a \to 6b$
$7b \to\cdot\ 8$	$a7 \to 7a$	$7a \to 7b$
$8b \to\cdot\ 9$	$a8 \to 8a$	$8a \to 8b$
$9b \to b0$	$a9 \to 9a$	$9a \to 9b$
$b \to\cdot\ 1$		$\to a$

Применим этот н. а. к пустому слову $R = \lambda$. Нетрудно понять, что на каждом шаге должна будет применяться самая последняя формула данной схемы. Получается бесконечный процесс: $\lambda \to a \to aa \to aaa \to aaaa \to \cdots$. Это означает, что к пустому слову данный алгоритм не применим.

Работа этого нормального алгоритма над словом $R = 499$ состоит из следующей последовательности слов: $499 \to a499 \to 4a99 \to 49a9 \to 499b \to 49b0 \to 4b00 \to 500$.

Таким образом, $s\ (499) = 500$.

Пример 4.14. Нормальный алгоритм, вычисляющий функцию $g(x) = 2x$ в десятичной системе счисления.

Решение:

В качестве алфавита для записи значений аргумента x и функции $g(x)$ возьмем алфавит $A = \{\lambda,\ 0,\ 1,\ 2,\ 3, \dots,\ 9\}$, а нормальный алгоритм будем строить в его расширении $B = A \cup \{a,\ b,\ c\}$.

Пусть w – заданное десятичное число. По аналогии с предыдущей задачей после перехода $w \to aw \to wa \to wb$ (работают второй и третий столбец подстановок схемы примера 13) нужно приступить к поразрядному умножению на 2. Для цифр 0, 1, 2, 3, 4 это выглядит так: $0b \to b0$, $1b \to b2$, $2b \to b4$, $3b \to b6$, $4b \to b8$. При удвоении остальных цифр приходится единицу переносить в следующий разряд. Для запоминания этого применяется дополнительная буква c расширенного алфавита: $5b \to c0$, $6b \to c2$, $7b \to c4$, $8b \to c6$, $9b \to c8$. Если мы к следующему разряду подошли с буквой b, то в

нем происходят только что описанные замены. Если же мы подошли с буквой c, то там выполняются следующие подстановки: $0c \to b1$, $1c \to b3$, $2c \to b5$, $3c \to b7$, $4c \to b9$, $5c \to c1$, $6c \to c3$, $7c \to c5$, $8c \to c7$, $9c \to 9c$.

После того как мы добрались от младшего разряда перерабатываемого слова до старшего, должен произойти останов. Если мы вышли за число с буквой b, то выполняется заключительная подстановка $b \to \cdot \lambda$. Если мы подошли с буквой c, то выполняется заключительная подстановка $c \to \cdot 1$.

Итак, схема нормального алгоритма представляет собой следующий упорядоченный набор подстановок (читается по столбцам):

$a0 \to 0a$	$0a \to 0b$	$0b \to b0$	$0c \to b1$	$b \to \cdot \lambda$
$a1 \to 1a$	$1a \to 1b$	$1b \to b2$	$1c \to b3$	$c \to \cdot 1$
$a2 \to 2a$	$2a \to 2b$	$2b \to b4$	$2c \to b5$	$\to a$
$a3 \to 3a$	$3a \to 3b$	$3b \to b6$	$3c \to b7$	
$a4 \to 4a$	$4a \to 4b$	$4b \to b8$	$4c \to b9$	
$a5 \to 5a$	$5a \to 5b$	$5b \to c0$	$5c \to c1$	
$a6 \to 6a$	$6a \to 6b$	$6b \to c2$	$6c \to c3$	
$a7 \to 7a$	$7a \to 7b$	$7b \to c4$	$7c \to c5$	
$a8 \to 8a$	$8a \to 8b$	$8b \to c6$	$8c \to c7$	
$a9 \to 9a$	$9a \to 9b$	$9b \to c8$	$9c \to c9$	

Пример 4.15.

Нормальный алгоритм, вычисляющий функцию $sum\ (x,\ y) = x + y$ в унарной системе, задается алфавитом $B = \{1, *\}$ и одной из следующих нормальных схем $\{* 1 \to \cdot$, $\{* 1 \to$, $\{1 * \to \cdot$, $\{1 * \to$, $\{1 * 1 \to 1$, $\{1 * 1 \to \cdot\ 1$.

Алфавит B пригоден для записи систем целых неотрицательных чисел. Так, если слова 111, 111111 представляют натуральные числа 2 и 5 соответственно, то слово 111*111111 в алфавите B представляет уже систему этих чисел. В данном примере нужно построить алгоритм, позволяющий по любому слову $1^{x+1} * 1^{y+1}$ в алфавите B получать слово 1^{x+y+1} в этом же алфавите. Требуемый алгоритм можно задать предписанием: «Сотри звездочку и одну единицу и остановись. Полученное слово и есть результат». Этот алгоритм является алгоритмом сложения системы целых неотрицательных чисел, перерабатывающим слово, обозначающее систему двух чисел, в слово,

обозначающее сумму этих чисел. Алгоритм можно задать с помощью одной из выше указанных подстановок, так как применение каждой из этих подстановок стирает звездочку и одну единицу. Процесс завершается либо в результате применения заключительной подстановки, либо в результате отсутствия вхождения левой части подстановки в полученном слове. Данный алгоритм применим ко всем словам вида $1^{x+1} * 1^{y+1}$, и результатом его работы будет слово 1^{x+y+1}.

Пример 4.16. Нормальный алгоритм, вычисляющий функцию $mod\ (x,\ y) = |x - y|$, задается алфавитом $B = \{1, *\}$ и следующей нормальной схемой: $\begin{cases} 1 * 1 \rightarrow * \\ * \rightarrow 1 \end{cases}$.

Алгоритм можно задать следующим предписанием:

1. Подставь вместо первого вхождения слова $1 * 1$ в данное слово $*$, переходи к указанию 2.
2. Если во вновь полученном слове есть вхождение слова $1 * 1$, то переходи к указанию 1, иначе – к указанию три.
3. Замени $*$ единицей (1 нужна для кодировки чисел), переходи к указанию 4.
4. Процесс вычисления завершился. Полученное слово является результатом.

Работа этого алгоритма над словом 111*111111 приводит к следующей последовательности слов: $111 * 111111 \rightarrow 11 * 11111 \rightarrow 1 * 1111 \rightarrow * 111 \rightarrow 1111$.

Получаем $mod\ (2,\ 5) = |2 - 5| = 3$.

Принцип нормализации Маркова: всякая эффективно вычислимая функция является нормально вычислимой, или всякий алгоритм представим в виде нормального алгоритма.

Математически доказать принцип нормализации невозможно, поскольку понятия произвольного алгоритма или эффективно вычислимой функции не являются строго определенными математическими понятиями.

Справедливость этого принципа основана на том, что все известные в настоящее время алгоритмы являются нормализуемыми, а способы композиции алгоритмов, позволяющие строить новые алгоритмы из уже известных, не выводят за пределы класса нормализуемых алгоритмов.

1.2. Основные способы композиции нормальных алгоритмов

1. Суперпозиция. При суперпозиции двух алгоритмов C и D выходное слово первого алгоритма C рассматривается как входное слово второго алгоритма D. Результат суперпозиции алгоритмов C и D можно представить в виде $H(p) = D \circ C = D(C(p))$. Суперпозиция может выполняться для любого конечного числа алгоритмов.

2. Объединение. *Объединением* алгоритмов C и D в одном и том же алфавите A называется алгоритм $H = CD$ в том же алфавите, преобразующий любое слово R, содержащееся в пересечении областей определения алгоритмов C и D в записанные рядом слова $C(R)$ и $D(R)$, на всех остальных входных словах этот алгоритм считается неопределенным.

3. Разветвление. *Разветвление* алгоритмов представляет собой композицию трех алгоритмов C, D и K. Обозначая результат этой композиции буквой H, будем считать, что область определения алгоритма H совпадает с пересечением областей определения всех трех алгоритмов C, D и K, а для любого слова R из этого пересечения $H(R) = C(R)$, если $K(R) = \lambda$, $H(R) = D(R)$, если $K(R) \neq \lambda$, где λ – пустая строка.

4. Повторение (итерация). *Итерация* представляет собой композицию двух алгоритмов C и D. Обозначая результат этой композиции через H, определим, что для любого входного слова R соответствующее ему выходное слово $H(R)$ получается в результате последовательного многократного применения алгоритма C до тех пор, пока не получится слово, перерабатываемое алгоритмом D в некоторое фиксированное слово.

Замечание. Все композиции нормальных алгоритмов приводят к нормальным алгоритмам.

Пример 4.17. *(суперпозиция алгоритмов).* Рассмотрим алгоритм, позволяющий по любому слову в алфавите $A = \{a,\ b\}$ определить, имеет ли оно четную длину.

Условимся считать, что этот алгоритм должен перерабатывать слова в алфавите $\{a, b\}$, имеющие четную длину (и только их), в пустое слово.

В таком случае нормальный алгоритм задается алфавитом $B = A$ и нормальной схемой

$$S_a: \begin{cases} a \to | \\ b \to | \\ | \quad | \to \lambda \end{cases}.$$

Этот алгоритм может быть задан в виде следующего предписания:

1. Вместо каждой буквы из алфавита $\{a,\ b\}$ запиши $|$, переходи к указанию 2.

2. Выясни, содержит ли полученное слово вхождение слова $||$. Если да, переходи к указанию 3, иначе – к указанию 4.

3. Выброси каждое вхождение слова $||$, переходи к указанию 4.

4. Сравни полученное слово с пустым. Если они совпадают, то переходи к указанию 5, иначе – к указанию 6.

5. Слово имеет четную длину. Переходи к указанию 7.

6. Слово имеет нечетную длину. Переходи к указанию семь.

7. Процесс вычисления остановить.

В работе рассмотренного алгоритма над исходным словом можно выделить два этапа: а) нахождение длины исходного слова;

б) распознавание четности длины слова.

Рассмотрим теперь два алгоритма: *C* – алгоритм, перерабатывающий каждое слово в алфавите $\{a,\ b\}$ в его длину. Алгоритм *C* задается нормальной схемой $\begin{cases} a \to | \\ b \to | \end{cases}$. *K* – алгоритм, распознающий по записи любого натурального числа в алфавите $\{\ |\ \}$, является ли оно четным. Алгоритм *D* задается

нормальной схемой $\{\ |\ \ |\to\lambda$. Работа описанного в этом примере алгоритма состоит в последовательном применении двух алгоритмов C и D таким образом, что исходным словом R для этого алгоритма является исходное слово R алгоритма C, а результат работы алгоритма C над этим словом слово R_C – является исходным словом для алгоритма D. Результат работы алгоритма D над словом R_C – слово R_D – есть результат работы рассматриваемого алгоритма над словом R.

Таким образом, алгоритм из примера 17 является суперпозицией алгоритмов $D \circ C$.

Пример 4.18. (*объединение алгоритмов*). Пусть заданы алфавит $A = \{a,\ b\}$, нормальные алгоритмы в том же алфавите, имеющие следующие нормальные схемы $C = \{ab \to ba$ и $D = \{ba \to ab$, и слово из области их пересечения $R = aba$. Тогда $C\ (aba) = baa$, $D\ (aba) = aab$, а объединение этих алгоритмов $H = CD$ даст результат $H(aba) = baaaab$.

Пример 4.19. (*разветвление алгоритмов*). Пусть заданы алфавит $A = \{a,\ b\}$, нормальные алгоритмы в том же алфавите, имеющие следующие нормальные схемы $C = \{ab \to ba$ и $D = \{ba \to ab$, и $K = \begin{cases} ab \to a \\ ba \to \lambda \end{cases}$.

Рассмотрим действие алгоритма H на слова $R = aba$ и $Q = bab$.

1) $C\ (aba) = baa$; $D(aba) = aab$; $K(aba) = aa$; $H(aba) = aab$.

2) $C\ (bab) = bba$; $D(bab) = abb$; $K(bab) = \lambda$; $H(bab) = bba$.

Пример 4.20. (*итерация алгоритмов*). Пусть заданы алфавит $A = \{a,\ b\}$, нормальные алгоритмы в том же алфавите, имеющие следующие нормальные схемы $C = \{ab \to ba$ и $D = \{bbbaa \to ab$.

Тогда $H(ababb) = ab$, так как $ababb \to baabb \to\ \to babab \to bbaab \to bbaba \to bbbaa \to ab$.

После пятикратного применения подстановки н. а. C получилось слово, перерабатываемое алгоритмом D в фиксированное слово ab.

Задачи

4.1. Нормальный алгоритм задан алфавитом $A = \{a, b\}$ и нормальной схемой $\begin{cases} ba \to ab \\ ab \to \end{cases}$.

Примените этот алгоритм к словам: 1) $bbaabab$; 2) $aabbaab$; 3) $abaabb$; 4) aaa; 5) $bbbb$.

Какое свойство слов распознается с помощью этого алгоритма? Укажите всевозможные результаты преобразования слов этим алгоритмом.

4.2. Нормальный алгоритм задан алфавитом $A = \{a,\ b\}$ и нормальной схемой $\begin{cases} ba \to ab \\ aa \to \\ bb \to \end{cases}$.

Примените этот алгоритм к словам:

1) $bbaaababa$; 2) $abbababab$; 3) $ababab$; 4) $baabbaba$.

Укажите всевозможные результаты преобразования слов этим алгоритмом.

4.3. Нормальный алгоритм задан алфавитом $A = \{a, b,\ |\}$ и нормальной схемой $\begin{cases} a \to\ | \\ b \to\ | \end{cases}$.

Примените этот алгоритм к словам: 1) $abaabbb$; 2) $bababbaa$; 3) aaa; 4) $bbbb$; 5) $aabbb|\ \ |$.

Что определяется для каждого исходного слова с помощью этого алгоритма?

4.4. Н. а., задан схемой (укажите алфавит н. а.):

1) $\begin{cases} \gamma aa \to \gamma \\ \gamma a \to \gamma \\ \gamma \to \cdot \\ \gamma b \to b\gamma a \\ \to \gamma \end{cases}$

Применим ли н. а., заданный схемой к словам: а) $aaaaa$; б) $ba^n b$; в) $bbbb$; г) $ab^n a$; д) к пустому слову ?

2) $\begin{cases} \gamma a \to a a \gamma \\ \gamma b \to b a \gamma \\ \gamma \to a \beta \\ a a a \beta \to \cdot \\ b \beta \to b \beta \\ \beta a \to a \beta \\ \beta b \to \gamma a \\ \to \gamma \end{cases}$ к словам: а) $aaaa$; б) $bbbbb$; в) $abab$; г) $bababa$?

3) $\begin{cases} \delta a \to \beta b \gamma \\ \delta b \to \gamma a \beta \\ \beta a \to \gamma b \delta \\ \beta b \to \delta a \gamma \\ \delta \beta \to \beta \delta \\ \delta \gamma \to \gamma \delta \\ \gamma \beta \to \beta \gamma \\ \delta \delta \to \delta \\ \beta \beta \to \beta \\ \gamma \delta \to \delta \gamma \\ \gamma \gamma \to \gamma \\ \beta \gamma \to \gamma \beta \\ \beta \delta \to \delta \beta \\ \to \delta \end{cases}$

к словам: а) $abab$; б) $baba$; в) a^n и b^n; г) к пустому слову?

4.5. Какую функцию вычисляет нормальный алгоритм:

а) $\lambda \to \cdot \lambda$; б) $\lambda \to \lambda$?

4.6. Постройте н.а., преобразующий каждое слово R в алфавите $A = \{a, b, c\}$ в слово $abcR$.

4.7. Разобрать пример 8 и записать нормальный алгоритм, вычисляющий словарную функцию $F(p) = pQ$ в алфавите $A = \{0,\ 1\}$.

4.8. Постройте н. а., преобразующий каждое слово в алфавите $A = \{a, b, c\}$ в слово abc.

4.9. Разобрать пример 9 и применить описанный в нем н.а. к следующим словам в алфавите $A = \{a, b, c\}$: а) $R = aabbcc$; б) $R = bacbacbac$; в)$R = ccabbc$;

4.10. Разобрать пример 10 и применить этот н. а. к следующим словам в алфавите $A = \{a, b, c\}$:

а) $R = aabbcc$; б) $R = bacbacbac$; в)$R = ccabbc$; г) $R = \lambda$.

4.11. Постройте н. а., преобразующий каждое слово в алфавите $A = \{a,\ b\}$ в:

1) пустое слово («аннулирующий» н. а.);

2) слово $baba$;

3) слово $x_1x_2 \ldots x_n$ в слово $x_2 \ldots x_nx_1$ (см. пример 10);

4) слово, буквы которого записываются в обратном порядке («переворачивающий» н. а.). Указание: см. пример 10.

4.12. Построить н. а., подсчитывающий число букв «*a*» в каждом слове, заданном в алфавите $A = \{a,\ b,\ c\}$.

4.13. Применить н. а. из примера 15 к переработке слов $1 * 1^{10}$ и $1^{10} * 111$. Что обозначают исходные слова и результаты?

4.14. Разобрать пример 15 и построить н. а., вычисляющий функцию:

1) $f(x_1,\ x_2,\ x_3) = x_1 + x_2 + x_3$;

2) $f(x_1,\ x_2, \ldots,\ x_n) = x_1 + x_2 + \cdots + x_n$.

4.15. Построить нормальные алгоритмы над алфавитом $A = \{0,\ 1\}$, вычисляющие следующие функции:

1)$f(x) = 2 + x$; 2)$f(x) = 2x$; 3)$f(x, y) = 2x + y$;

4)$f(x) = \begin{cases} 1, & \textit{если } x \textit{ делится на } 2 \\ 0, & \textit{если } x \textit{ не делится на } 2 \end{cases}$;

5)$f(x) = \begin{cases} 1, & \textit{если } x \textit{ делится на } 3 \\ 0, & \textit{если } x \textit{ не делится на } 3 \end{cases}$;

6) $f(x) = 2x$; 7) $f(x) = \frac{x}{2}$;

8) $f(x) = \left[\frac{x}{3}\right]$; 9)$f(x) = \frac{x}{3}$;

10)$f\ (x,\ y,\ z) = I_1^3(x,\ y,\ z) = x$;

11)$f(x,\ y,\ z) = I_2^3(x,\ y,\ z) = y$;

12)$f(x,\ y,\ z) = I_3^3(x,\ y,\ z) = z$;

13)$f(x,\ y,\ z) = 2x + 3y + 5z$; 14) $f(x) = |x - 5|$;

15)$f(x,\ y,\ z,\ t) = 3x + 5z + 1$;

16)$f(x,\ y,\ z,\ t) = t + 4y + 2$; 17)$f(x) = max\ \ (x,\ 5)$;

18)$f(x) = x \dot{-} 1 = \begin{cases} x-1, & \textit{если } x \geq 1 \\ 0, & \textit{если } x = 0 \end{cases}$;

19)$f(x) = x - 1 = \begin{cases} x-1, & \textit{если } x \geq 1 \\ \textit{не определена}, & x = 0 \end{cases}$;

20)$sg(x) = \begin{cases} 1, \textit{ если } x \geq 1 \\ 0, \textit{ если } x = 0 \end{cases}$; 21)$\overline{sg\ (x)} = \begin{cases} 1, \textit{ если} x = 0 \\ 0, \textit{ если} x \geq 1 \end{cases}$;

22)$sg(x-3)$; 23)$sg(x \dot{-} 3)$;

24)$\overline{sg}\ (x-3)$; 25)$\overline{sg}\ (x \dot{-} 3)$;

26)$f(x) = x \dot{-} 3$; 27)$f(x) = x - 3$;

28)$f(x) = 2x - 3$; 29)$f(x) = 2x \dot{-} 3$;

30)$f(x) = sg\ (2x-3)$; 31)$f(x) = sg\ (2x \dot{-} 3)$;

32)$\overline{sg}(2x-3)$; 33) $\overline{sg}(2x \dot{-} 3)$;

34) $f(x,\ y) = x \dot{-} y = \begin{cases} x-y, & \textit{если } x \geq y \\ 0, & \textit{если } x \langle\ y \end{cases}$;

35) $f(x) = x^2$; 36)$f(x,y) = x \cdot y$;

37) $max\ \ (x,y)$; 38) $f(x,\ y) = \frac{x}{2} + y$;

39) $f(x) = \left[\frac{1}{x}\right] = \begin{cases} 1, & \textit{если } x = 1 \\ 0, & \textit{если } x \geq 2 \\ \textit{не определено}, & \textit{если} x = 0 \end{cases}$;

40) $f(x) = \frac{2}{4-x}$; 41) $f(x) = 2^{1-x}$.

4.16. Построить нормальные алгоритмы, вычисляющие следующие функции в десятичной системе счисления:

1)$f(x) = x + 2$; 2) $f(x) = x + 3$;

3) $f(x) = x + 5$; 4)$f(x) = x + 7$;

5)$f(x) = x - 1$; 6) $f(x) = x - 2$;

7) $f(x) = x - 5$; 8)$f(x) = x - 7$;

9)$f(x) = x \dot{-} 1$; 10) $f(x) = x \dot{-} 2$;

11) $f(x) = 10x$; 12)$f(x) = 3x$.

4.17. Построить нормальный алгоритм подсчета букв «a» в любом слове, записанном в алфавите $A = \{a,\ b\}$.

4.18. Построить нормальный алгоритм, подсчитывающий количество букв «b» в слове, записанном в алфавите $A = \{a,\ b\}$ и содержащем более одной буквы «b», иначе заменить исходное слово на «bbb».

4.19. Написать н. а., преобразующий любое слово в алфавите $A = \{a,\ b\}$, содержащее хотя бы две буквы «a», в слово «$baabab$», иначе слово стереть.

4.20. Написать н. а., преобразующий любое слово в алфавите $A = \{a,\ b\}$, содержащее хотя бы две буквы «a» в слово «$baabab$», иначе оставить слово без изменения.

4.21. Построить нормальный алгоритм, преобразующий любое слово в алфавите $A = \{a,\ b\}$, содержащее хотя бы две буквы «b» в слово «abb», иначе в слово, полученное из исходного заменой «a» на «b» и наоборот.

4.22. Построить нормальный алгоритм, который вычисляет предикат симметрии $S\ (x)$: для любого слова $x = x_1 x_2 \ldots x_n$ в алфавите $\{a,\ b\}$

$$S\ (x) = \begin{cases} 1, & \textit{если} x_i = x_{n-i+1} \quad \textit{для всех} \quad i = 1,\ 2, \ldots,\ n \\ 0, & \textit{в противном случае} \end{cases}$$

Глоссарий

1. Применимость (неприменимость) алгоритма к исходному данному: алгоритм применим, если процесс заканчивается получением результата. В противном случае процесс может оборваться (безрезультатная остановка) или не закончиться (продолжаться бесконечно).

2. Массовая алгоритмическая проблема – бесконечная серия однотипных единичных задач, требующая единого метода (алгоритма решения). Если такого алгоритма не существует, данная массовая проблема называется неразрешимой.

3. Вычислимая (эффективно вычислимая) функция – функция, для вычисления которой существует алгоритм.

4. Операция суперпозиции функций – функция

$F(x_1,\ x_2, \dots,\ x_n) = \phi\ (f_1(x_1, \dots,\ x_n), f_2(x_1, \dots, x_n), \dots f_m(x_1, \dots, x_n))$

называется *суперпозицией функций* ϕ и $f_1, \dots, f_m$ и обозначается через

$F = S\ (\phi, f_1, \dots, f_m)$, причем функция F определена на наборе $\tilde{\alpha}^n = (a_1, a_2, \dots, a_n)$ тогда и только тогда, когда каждая функция $f_i\ \ (i = 1, \dots m)$ определена на наборе $\tilde{\alpha}^n$ и, кроме того, функция ϕ определена на наборе $(f_1(\tilde{\alpha}^n), \dots, f_m(\tilde{\alpha}^n))$; в этом случае $F\ (\tilde{\alpha}^n) = \phi\ (f_1(\tilde{\alpha}^n), f_2(\tilde{\alpha}^n), \dots, f_m(\tilde{\alpha}^n))$.

5. Схема примитивной рекурсии: функция f от $(n + 1)$ переменных получается применением схемы примитивной рекурсии R к функциям g от n переменных и h от $(n + 2)$ переменных (обозначение $f = R(g, h)$), если для всех $x_i \in N_0$:

1) $f(x_1,\ x_2, \dots,\ x_n,\ 0) = g(x_1, \dots, x_n)$;

2) $f(x_1,\ x_2, \dots,\ x_n,\ k + 1) = h(x_1, \dots, x_n,\ k,\ f(x_1, \dots, x_n,\ k)), k \in N_0$.

6. Примитивно рекурсивная функция (п/р) – функция, которая может быть получена из базисных (исходных) функций: $o(x) = 0$ (константы 0), $s(x) = x + 1$ (функции следования), $I_k^n(x_1, \dots, x_k, \dots, x_n) = x_k$ (функций выбора) применением конечного числа операций суперпозиции и примитивной рекурсии.

7. Оператор минимизации (μ -оператор). Говорят, что функция $g(x_1,\ x_2, \dots,\ x_n)$ получена из функций $f(x_1,\ x_2, \dots,\ x_n)$ с помощью *операции минимизации или с помощью μ-оператора* (обозначение $g = M(f)$ или $g(x_1,\ x_2, \dots,\ x_n) = \mu_z(f(x_1,\ x_2, \dots,\ x_{n-1},\ z) = x_n)$), если выполнено условие $g(x_1,\ x_2, \dots,\ x_n)$ определена и равна z тогда и только тогда, когда $f(x_1,\ x_2, \dots,\ x_{n-1},\ 0)$, $f(x_1,\ x_2, \dots,\ x_{n-1},\ 1), \dots,$ $f(x_1,\ x_2, \dots,\ x_{n-1},\ z-1)$ определены и не равны x_n, а $f(x_1,\ x_2, \dots,\ x_{n-1},\ z) = x_n$.

8. Частично - рекурсивная функция (ч/р) – функция, которая может быть получена из исходных базисных функций применением конечного числа операторов S (суперпозиции), R (примитивной рекурсии) и μ-операторов.

9. Общерекурсивная функция – ч/р функция, если она определена всюду на N_0.

10. Тезис Черча: всякая эффективно (алгоритмически) вычислимая функция является частично рекурсивной.

11. Программа МТ – система команд вида $q_i a_j \to q_k a_l S$, где q_i (соответственно a_j) – символы внутреннего (соответственно внешнего) алфавитов, S– символ сдвига указателя МТ: L (влево), R (вправо), E (на месте); $q_i\ ,\ q_k \in Q, (1 \le i \le m, 0 \le k \le m),$ $a_j,\ a_l \in A.$

12. Машина Тьюринга (МТ) – система, задаваемая тройкой $T = (A,\ Q,\ P)$, которая удовлетворяет следующим условиям:

1) множества $A = \{a_1,\ a_2, \dots,\ \boldsymbol{a_n}\}$ – внешний алфавит МТ и $\boldsymbol{Q} = \{\boldsymbol{q_0},\ \boldsymbol{q_1}, \dots,\ \boldsymbol{q_m}\}$ – внутренний алфавит состояний управляющего устройства конечны, не пересекаются и не содержат букв L, R, E;

2) пустой символ $a_0 \in A$; начальное и заключительное состояния $q_1,\ q_0 \in Q$;

3) P – такая программа с внешним алфавитом A и внутренним алфавитом Q, что

а) не существует в P двух различных команд с одинаковыми левыми частями;

b) q_0 не входит в левую часть ни одной команды из *P*.

4) МТ – воображаемое устройство, состоящее из управляющего устройства, реализующего функционирование МТ в дискретном времени по программе *P*; ленты, разделенной на ячейки; читающего-пишущего указателя.

13. Конфигурация МТ – машинное слово *K* вида Uq_iV, где U – слово на ленте левее указателя, V – слово правее слова U, начиная с символа, находящегося в той ячейке, на которую указывает указатель.

14. Протокол работы МТ – детерминированная (конечная или бесконечная) последовательность конфигураций K_{i_1}, K_{i_2}, ..., такая, что T: $K_{i_r} \rightarrow K_{i_{r+1}}$, т. е. при выполнении некоторой команды машины T из конфигурации K_{i_r} получается конфигурация $K_{i_{r+1}}$.

15. Машина Т применима, к слову *R* – начав работу на слове R, МТ T остановится через конечное число шагов; если при этом получается слово Q, то пишут $T(R) = Q$. Если *T* не останавливается, то *T* не применима к слову R.

16. Машина Тьюринга *T* правильно вычисляет функцию $f(x_1,\ x_2, \ldots,\ x_n)$, если:

1) в случае, когда $f(m_1,\ m_2, \ldots,\ m_n) = m$ определено, МТ, начав работу с левой единицы кода $1^{m_1+1} * 1^{m_2+1} * \ldots * 1^{m_n+1}$, останавливается, и указатель машины в заключительной конфигурации обозревает левую единицу кода 1^{m+1}.

2) в случае, когда $f(m_1,\ m_2, \ldots,\ m_n)$ не определено, МТ, начав работу с левой единицы кода $1^{m_1+1} * 1^{m_2+1} * \ldots * 1^{m_n+1}$, не останавливается.

Функция называется правильно вычислимой, если есть МТ, которая ее правильно вычисляет.

17. Композиция машин Тьюринга T_1 и T_2: для двух МТ $T_1 = (A_1,\ Q_1,\ P_1)$ и $T_2 = (A_2,\ Q_2,\ P_2)$ с внешними алфавитами A_1 и A_2, алфавитами состояний $Q_1 = \{q_0',\ q_1', \ldots,\ q_r'\}$ и $Q_2 = \{q_0'',\ q_1'', \ldots,\ q_s''\}$, композиция – это МТ $T_1\mathrm{T}_2$=(A, Q, P), где $A = A_1 \cup A_2$, $Q = \{q_0' = q_1'',\ q_1', \ldots,\ q_r',\ q_1'', \ldots,\ q_s''\}$, $P = P_1' \cup$

P_2 и P_1' получается из P_1 заменой q_0' на q_1'' (т. е. вместо окончания работы T_1 начинает работать T_2).

18. Теоремы об эквивалентности МТ и ч/р функции: 1) теорема Тьюринга: все частично рекурсивные функции правильно вычислимы;

2) всякая функция, вычислимая машиной Тьюринга, частично рекурсивна.

19. Универсальная машина Тьюринга – МТ, которая по заданным на ленте программе любой м/т T и некоторой исходной ситуации X выполняет преобразование X в результат применения T к слову X. Универсальную МТ можно также назвать интерпретатором.

20. Тезис Тьюринга: всякий алгоритм может быть реализован машиной Тьюринга.

21. Марковскими подстановками называются выражения вида: $P \to Q$ (обычная формула подстановки), $P \to\cdot\ Q$ (заключительная формула подстановки), где P и Q – слова в алфавите A, не содержащем в качестве букв знаки $\to$ и $\cdot$. Марковские подстановки представляют собой операции над словами в данном алфавите A. Каждая операция задается с помощью упорядоченной пары слов $(P,\ Q)$ и состоит в том, что в заданном слове R находят *первое* вхождение слова P (если таковое имеется) и, не изменяя остальных частей слова R, заменяют в нем это вхождение словом Q. Полученное слово называется результатом применения марковской подстановки к слову R. Если же первого вхождения слова P в слове R нет (и, следовательно, нет вообще ни одного вхождения P в R), то считается, что марковская подстановка не применима к слову R.

22. Нормальной схемой S_a над алфавитом A называется произвольная конечная непустая упорядоченная последовательность марковских подстановок.

23. Нормальным алгоритмом Маркова над алфавитом A называется следующее правило построения последовательности $R \Rightarrow R_1 \Rightarrow \cdots \Rightarrow R_i \Rightarrow$

$R_{i+1} \Rightarrow \cdots$ слов в алфавите $B \supseteq A$, исходя из данного слова R в алфавите A. Пусть для некоторого $i \geq 0$ слово R_i построено и процесс построения рассматриваемой последовательности еще не завершился. Если при этом в схеме нормального алгоритма нет формул, левые части которых входили бы в R_i, то R_{i+1} полагают равным R_i, и процесс построения последовательности считается завершившимся. Если же в схеме имеются формулы с левыми частями, входящими в R_i, то в качестве R_{i+1} берется результат марковской подстановки правой части первой из таких формул вместо первого вхождения ее левой части в слово R_i; процесс построения последовательности считается завершившимся, если на данном шаге была применена формула заключительной подстановки, и продолжающимся – в противном случае.

24. Нормальный алгоритм называется **применимым к слову** R, если процесс построения последовательности слов обрывается $R \Rightarrow R_1 \Rightarrow \cdots \Rightarrow R_{m-1} \Rightarrow R_m$. Последний член последовательности R_m называется **результатом** применения нормального алгоритма к слову R. Говорят, что нормальный алгоритм **перерабатывает** слово R в слово R_m. Нормальный алгоритм **не применим** к слову R, если последовательность слов бесконечна.

25. Функция *f*, заданная на некотором множестве слов алфавита *A*, называется **нормально вычислимой**, если найдется такое расширение *B* данного алфавита *A* ($A \subseteq B$) и такой нормальный алгоритм в *B*, что каждое слово R (в алфавите *A*) из области определения функции f этот алгоритм перерабатывает в слово $f(R)$.

26. Принцип нормализации Маркова: всякая эффективно вычислимая функция является нормально вычислимой.

Литература

1. Гаврилов Г. П., Сапоженко А. А. Задачи и упражнения по дискретной математике. – М., 2017.

2. Игошин, В. И. Сборник задач по математической логике и теории алгоритмов: учеб. пособие/ В.И. Игошин. – Москва : КУРС: ИНФРА-М, 2017. – 392 с. – (Бакалавриат). – ISBN 978-5-16-103684-6. – Текст : электронный. – URL: https://znanium.com/catalog/product/524332

3. Клини С. Функциональные системы в дискретной математике: Пер. с англ. – М., 1973.

4. Лавров, И. А. Задачи по теории множеств, математической логике и теории алгоритмов : учебник / И. А. Лавров, Л. Л. Максимова. – 5-е изд., испр. – Москва : ФИЗМАТЛИТ, 2002. – 256 с. – ISBN 5-9221-0026-2. – Текст: электронный// Лань: электронно-библиотечная система. – URL: https://e.lanbook.com/book/2242.

5. Мальцев А. И. Алгоритмы и рекурсивные функции. – М., 1986.

6. Роджерс Х. Теория рекурсивных функций и эффективная вычислимость: пер. с англ. – М., 1972.

Прокопенко Наталья Юрьевна

ФУНКЦИОНАЛЬНЫЕ СИСТЕМЫ В ДИСКРЕТНОЙ МАТЕМАТИКЕ

Учебное пособие

Федеральное государственное бюджетное образовательное учреждение высшего образования «Нижегородский государственный архитектурно-строительный университет»
603950, Нижний Новгород, ул. Ильинская, 65.
http://www. nngasu.ru, srec@nngasu.ru